Spline-Based Motion Vector Encoding Scheme

by

Parnia Farokhian

A thesis submitted to the

Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

August, 2012

Abstract

We present a new motion vector encoding scheme based on a curve fitting algorithm; the motion vectors of collocated blocks in a video sequence are represented by a set of keypoints that are the coefficients of the best fitted curve into the motion vectors. Motion vectors are mapped into four different categories; each corresponding to the condition for recovering each motion vector from the curve losslessly. Using a proposed adaptive motion estimation method, rather than selecting the motion vector corresponding to the block of minimum residual energy, a set of motion vectors is chosen such that each candidate set contains motion vectors of residual blocks of a number of bits less than a threshold. We utilize a rate-distortion technique to estimate the number of bits per candidate residual block to avoid computational complexity for selecting the best key point for the curve in terms of residual data. Experimental results show a significant bitrate reduction up to 43% for encoding the motion vector data for the curves' and block residuals in comparison to the H.264/AVC codec.

Acknowledgments

I would like thank my supervisor, Professor Chris Joslin, for his assistance during the course of this research. His advice and suggestions have consistently helped to steer this work in a productive direction.

I have also been privileged enough to have supportive and motivating colleagues and friends. Two of my most supportive colleagues have been, Omar Hesham and Sina Firouzi. I couldn't have made it without you.

Last but not least I would also like to extend my deepest gratitude to Mohammad Shahsavari Goughari. Without his encouragement and support, I would not have a chance to be at Carleton University. Finally, I would like to thank my parents, Mehri Rasekh and Keykhosro Farokhian for all her invaluable support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increasing volume of communication data such as video and image signals carried over the transmission and storage environments have been making the field of image and video compression a very active throughout the past 20 years. Data compression is referred to as encoding, which involves transforming a series of bits into a new set of code words that conveys the same information, but in a smaller size. The compression process may be implemented by removing redundancy from the information-carrying signal. In a lossless compression system the original signal can be perfectly reconstructed at the receiver by removing the statistical redundancy, however, this results in only a modest amount of compression for image and video signals. Lossy video compression schemes that are based on removing temporal and/or spatial redundancy are more practical in achieving greater compression at the expense of a certain amount of information loss (distortion) such that the decoded signal is not identical to the original. Therefore the goal of a video encoder is to achieve efficient data compression while minimizing the distortion generated by the compression process. A typical video encoder consists of three main functional units: a temporal model, a spatial model and an entropy

encoder, as shown in Figure 1.1. The input to the temporal model is an uncompressed video sequence. The temporal model exploits the temporal redundancy between adjacent frames (or the frames captured at around the same time) to reduce the video data. A set of model parameters containing typically motion vectors and header data, describing the motion of the video objects (e.g. blocks of pixels) and a residual frame (created by subtracting the prediction from the actual current frame) are the output of the temporal model unit. The residual frames are the input to the spatial model which makes use of similarities between neighbouring samples in the residual frame by applying a transform to the residual samples and quantizing the results.

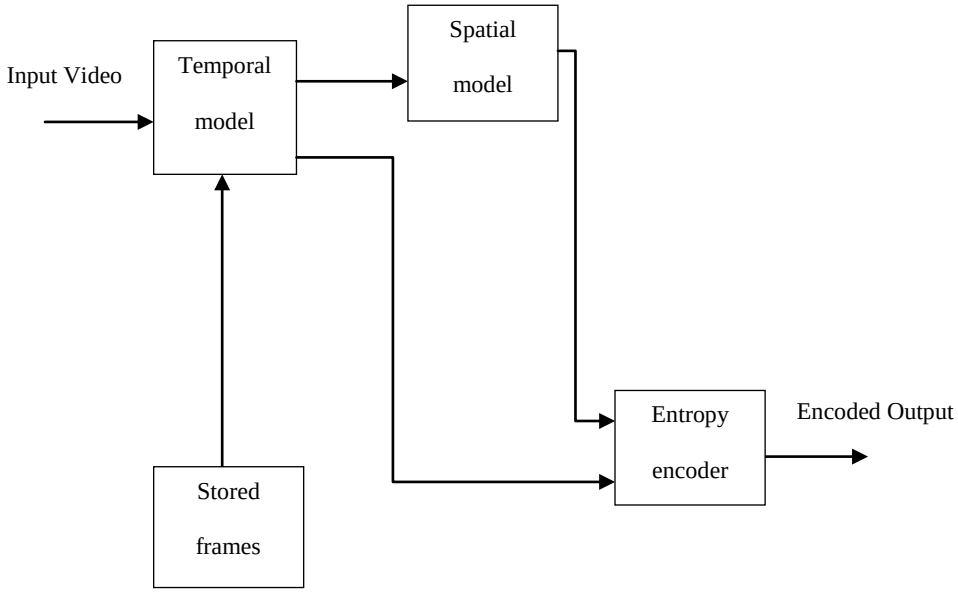


Figure 1.1: Video Encoder Block Diagram

The transform maps the samples into a set of transform coefficients. The coefficients are quantized to remove insignificant values (e.g. high frequency details), leaving a small number of significant coefficients that provide a more compact representation of the residual frame. The parameters of the temporal model (typically motion vectors) and the spatial model (coefficients)

are compressed by the entropy encoder. This removes statistical redundancy in the data (for example, representing commonly-occurring vectors and coefficients by shorter binary codes) and produces a compressed bit stream or file that may be transmitted and/or stored. Therefore, a compressed sequence consists of three main parts; coded motion vector parameters, coded residual coefficients, and header information. The number of bits required to encode each part of the video bitstream is affected by different factors which depend on the adjustments of the three model units. In particular motion vectors form a significant portion of the bitstream especially when trying to achieve a lower amount of distortion; in this case a more precise temporal model is applied for generating smaller residual data by using smaller block sizes. Therefore, if the number of bits for representing the motion vectors is decreased, while maintaining the performance of the temporal model, the compression efficiency can be increased significantly.

The motion vectors of collocated blocks in consecutive frames of a video may produce a motion field with the potential of being represented by a curve in a more compact way than representing each discrete motion vector value separately. However, representing a sequence of motion vectors belonging to collocated blocks along the video sequence by curve coefficients will only result in data compression if the number of coefficients is less than the number of actual motion vectors. Therefore, based on the curve fitting process the smoother the motion vector field is the lower the number of curve coefficients are needed to represent the motion vectors. Motivated by the fact that, smooth motions along the video sequence may be generated by either objects moving across the screen, or cameras moving in a controlled manner (as is popular in TV and cinema), we propose a lossless encoding technique for preprocessing motion vectors via a curve fitting algorithm which results in the reduction of the bitrate. However, in order to generalize the

curve fitting[1] algorithm to variety of motion types not only necessarily homogenous motions, an adaptive motion estimation process is proposed in this work such that the selected motion vectors finally results in a minimum number of curve coefficients while maintaining the encoding of the residual data.

1.1 Temporal Model and Regulating Factors of Motion Vector Bitrate

The temporal model operates as a motion compensated-estimation [1] process that in its simplest form the previous frame is used as the predictor, for the current frame and the residual, as shown in Figure 1.2, is formed by subtracting the predictor from the current frame.

(a) Predictor (b) Current frame (c) Residual

Figure 1.2: Two Adjacent Frames and the Residual Frame.

A practical and widely-used method of motion compensation is to compensate for the movement of a rectangular section or "block" of the current frame. The following procedure is carried out for each block of M × N samples in the current frame where M and N are the number of horizontally and vertically situated pixels per block width and length respectively. For each M×N block in the current frame the best matching block is found in the reference frame within a predefined search window which is usually centred at the same spatial address of the current block (Figure 1.3). The energy of the residual block is usually used as a measurement of similarity between the current block and the candidate matching blocks in the search window.

[1] Curve fitting is the process of finding a curve which is best fitted to set of data points. The curve fitting algorithm will be explained in more detail in Chapter 3.

The best matching block among the candidates is the block which results in minimum energy residual block. The energy of the residual block may be defined as Sum of Absolute Differences (SAD) or Mean Square Error (MSE), which are the most popular energy definitions. The chosen candidate region (best matching block) becomes the predictor for the current M × N block and is subtracted from the current block to form a residual M × N block (motion compensation). The spatial offset of a block in the current picture to the prediction block in the reference picture is determined by a 2-dimensional motion vector which is used to reconstruct the current block from the spatially processed residual block at the decoder side.

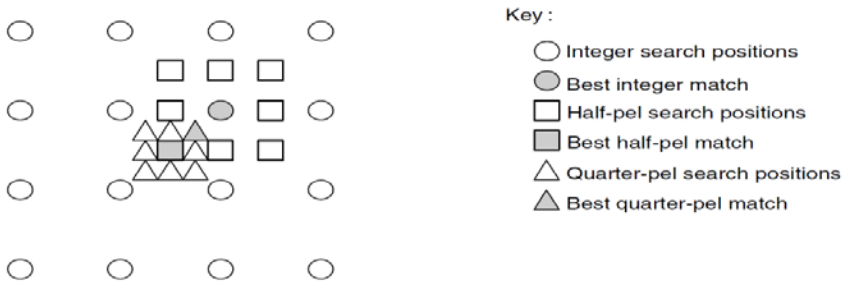
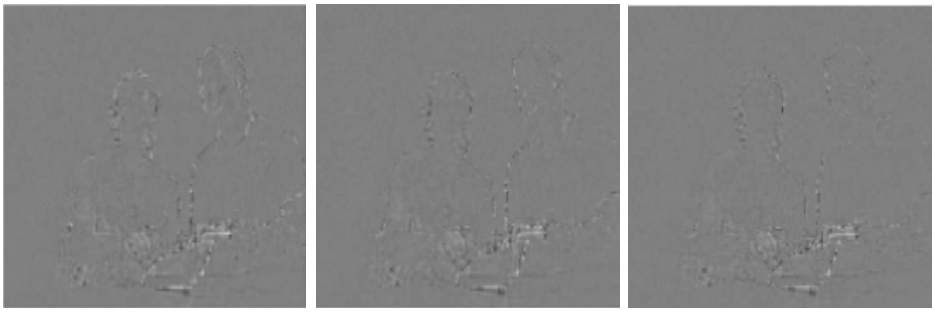


Figure 1.3: Fractional motion estimation.

There are many variations on the basic motion estimation and compensation process. The reference frame may be a previous frame (in temporal order), a future frame or a combination of predictions from two or more previously encoded frames. It is necessary to encode the reference frame before the current frame (i.e. frames must be encoded out of order). Where there is a significant change between the reference and current frames (for example, a scene change), it may be more efficient to encode the macroblock without motion compensation and so an encoder may choose intra mode (encoding without motion compensation) or inter mode (encoding with motion compensated prediction) for each macroblock. A macroblock of 16×16 pixels may contain more than one moving object, especially when the macroblock belongs to the detailed parts of the frame, so choosing a smaller block size and/or variable block size for motion

estimation and compensation [2] may be more efficient in terms of reducing the residual block data. However, variable block size motion estimation requires encoding the header information to signal the macroblock mode (the way macroblock is divided into smaller blocks) plus the motion vector for each sub-macroblock which itself results in a large amount of motion data. On the other hand, fixed block size motion estimation with a smaller block size, compared to the macroblock, is not an efficient method for areas in the frame that contain homogenous texture such as backgrounds. Therefore, a video coding algorithm should make a balance between the number of bits and the accuracy of the motion estimation process. Sub-pixel motion compensation [3] is another option when a better prediction may be formed by predicting from interpolated sample positions in the reference frame. Figure 1.3 shows the concept of 'quarter-pixel' motion estimation. In the first stage, motion estimation finds the best match on the integer sample grid (circles). The encoder searches the half-sample positions immediately next to this best match (squares) to see whether the match can be improved and if required, the quarter-sample positions next to the best half-sample position (triangles) are then searched. The final match (at an integer, half- or quarter-sample position) is subtracted from the current block or macroblock. The residual data shown in Figure 1.4a and Figure 1.4b are produced using an integer pixel and half pixel motion estimation systems respectively. This approach may be extended further by interpolation onto a quarter-sample grid to give a still smaller residual (Figure 1.4c). In general, 'finer' interpolation provides better motion compensation performance (a smaller residual) at the expense of increased complexity. The performance gain tends to diminish as the interpolation steps increase. Half-sample interpolation gives a significant gain over integer-sample motion compensation, quarter-sample interpolation gives a moderate further

improvement, and eighth-sample interpolation gives a small further improvement again and so on.



(a) integer (b) half-pixel (c) quarter-pixel

Figure 1.4: 4×4 residual blocks compensation.

1.2 Spatial Model and Regulating Factors for Residual Block Bitrate

The obvious problem with single temporal prediction is that a lot of energy remains in the residual frame and this means that there is still a significant amount of information to compress after temporal prediction. The function of the spatial model is to decorrelate the residual data further and to convert it into a form that can be efficiently compressed using an entropy coder. Practical spatial models typically have three main components, transformation, quantisation and reordering. The purpose of the transform stage in an image or video codec[2] is to convert image or motion-compensated residual data into another domain (the transform domain) in which samples are less correlated. In the transform domain, data is divided into vision-sensitive data (low-frequency components) and vision-insensitive data (high frequency components). Many transforms have been proposed for image and video compression and the most popular transforms tend to fall into two categories; block-based and image-based (such as Discrete Wavelet Transforms) An example of block-based transforms includes the ever-popular Discrete Cosine Transform (DCT) scheme which is a fundamental component of many image and video

[2] A video codec is a device or software that enables video compression or decompression for digital video

compression standards including JPEG [4], H.263 [5], MPEG [6], MPEG-2 [7], and MPEG-4 [8].The block-based transforms operate on square blocks of M×N image or residual samples and hence the image is processed into units of a block. Block transforms have low memory requirements and are well-suited to the compression of block-based motion compensation residuals but tend to suffer from artefacts at block edges due to the correlation among spatially adjacent blocks[3]. The DCT features in MPEG-4 Visual and a variant of the DCT is incorporated in H.264 [8].

1.3 Motivation and Problem Description

The increasing popularity of the use of motion estimation schemes, such as choosing smaller block sizes and variable block sizes (as described in Section 1.1), has led to an increase in the proportion of motion vectors (MV) data in the bitstream. This, coupled with the significant proportion of the motion vectors in the bitstream in low bitrate communication (as seen in Section 1.2), means that if the number of bits required for representing motion data is decreased while maintaining the performance of the motion compensation, the compression efficiency can be increased proportionally. To minimize the MV data, various MV encoding methods have been so far proposed in literature that are mostly based on applying predictive coding schemes before entropy encoding of motion vectors. The predictive preprocessing schemes make use of redundancy between motion vectors of temporally and/or spatially neighboring blocks to extract a smaller motion data from the motion vectors of the video sequence. For instance H.264/AVC standard applies a predictive coding method by using predictive motion vector (PMV) which is calculated by the median of three spatially neighboring MVs. The median PMV is effective in

[3] Tow general approaches have been proposed in literature to reduce the blocking artifact (blockiness). The first approach is based on dealing with blockiness at the encoder side ([9]-[10]) and the second one is post-processing the decoded video signal at the decoder side by improving the visual quality ([11]-[13]).

reducing the MV data, since neighboring blocks tend to move in a similar manner, and thus the error between the actual movement and the prediction is usually small.

An obvious challenge that arises from previously defined predictive techniques that utilise the spatial redundancy between motion vectors, is that the neighboring blocks do not always belong to a single object or they may belong to a single object with a complex motion so that it might turn out the neighboring blocks in a video frame do not move with the same rate and velocity. Therefore, extracting motion data out of motion vectors of neighboring blocks does not necessarily lead to decreasing the bitrate. Temporal predictive techniques also do not contribute to the improvement of the motion vector bitrate for the video sequences containing scenes changes.

In this work a new PMV concept is introduced which does not depend on a possible redundancy between neighboring motion vectors. Despite previous schemes the proposed method in this thesis is based on the true motion that exists in the video sequence as the new PMVs are a subset of motion vectors, not only a set of motion vector derivatives (e.g. the median predictor in H.264). The proposed method is based on a curve fitting algorithm such that the curve control points (new PMVs) and some header information are sufficient for recovering the motion vectors at the decoder side from the curve's formula. The concept behind fitting motion vectors to a curve is that as long as a curve is relatively smooth data can be represented by a set of control points (that are used to determine the curve itself) in a much more compact way than labelling each discrete point. As a great deal of motion in video is from either objects moving across the screen, or cameras moving in a controlled manner, smooth curves are easy to come by if we were to map the motion vectors to a curve. In order to generalize the method for the video sequences of not only smooth motions an adaptive motion estimation system is proposed such that the

output motion vectors are fitted to a curve with a minimum number of control points in addition compression efficiency increases by restricting the number of candidate matching blocks by a residual block rate control model (see Chapter 4).

1.4 Research Contributions

In order to achieve the objectives described, different contributions are proposed in this research:

a) *An automated curve fitting algorithm by which the minimum motion data bitrate for reconstructing motion vectors are generated.*

We proposed a new motion vector encoding technique based on a curve fitting algorithm which leads to saving bitrate up to amount of 42.7% in comparison with H.264 standard. The main difference between the new motion vectors encoding scheme and previously proposed methods is that in the previous methods for any inter-mode encoded block corresponding to a non-skipped macroblock a single 2-dimensional motion vector plus macroblock modes (in the case of variable block size motion estimation) are encoded in the bit stream, however in the new method only the motion vectors which are recognised as the curve control points are encoded and the rest of motion vectors (non-control point motion vectors) are recovered from the curves using the control points and the header information that contains a small amount of data.

b) *Increasing the number of skipped blocks*

A new skipped block definition is presented for which, similar to the original skipped macroblock definition, no motion information is encoded. It will be shown that the number skipped macroblocks increases as the new skipped blocks are the blocks of zero residual energy that have non-control point motion vectors while based on the general skipped macroblock definition the blocks of zero residual energy and zero predictive motion vector are recognized as skipped macroblocks. The motion vector of the new skipped blocks is reconstructed from the

curves while the motion vectors of the originally defined skipped macroblocks are estimated from the motion vectors of previously decoded blocks.

c) *Determining bitrate threshold for selecting matching block based on the motion type*

In order to optimise the encoding of the motion vector curve, in conjunction with the resulting block residual, we have proposed a novel update to the rate control model that takes into consideration both elements based on the thresholds in order to obtain the lowest bitrate.

1.5 Thesis Organization

This document is organized as follows: Chapter 2 provides a review of existing work on motion vector pre-encoding techniques in block-based motion compensated hybrid video coding, along with an overview of motion vector entropy encoding methods. Chapter 3 introduces the proposed motion vector encoding scheme that is structured as a 3-step algorithm containing our proposed adaptive motion estimation system, curve fitting algorithm, and entropy encoding process. Chapter 4 describes a new adaptive motion estimation process by first describing the rate control concept and then determining the bitrate threshold for selecting candidate blocks. In Chapter 5, the compression performance of the proposed motion vector encoder is evaluated in detail via simulation of the method using four standard test video sequences under different test conditions. Finally, Chapter 6 concludes the thesis and discusses potential future research directions.

Chapter 2

Literature Review

In order to reduce the number of bits required to transmit motion vectors, so far, typically two different aspects of data compression have been considered in literature. The first aspect is based on decreasing the entropy of the motion vectors which itself is divided into two sub-procedures; lossy techniques which are regarded as motion vector quantization methods and lossless methods that are more widespread and are based on generating predictive motion vectors (PMVs) out of motion vectors. All the pre-processing methods aim to reduce the motion data via removing the redundancy between motion vector values while the second aspect of motion vector compression, entropy encoding scheme, reduces the information by removing statistical redundancy between the generated motion data from the entropy reduction process.

2.1 Motion Vector Pre-Encoding Techniques

2.1.1 Lossy Motion Vector Pre-encoding Techniques

Lossy encoding methods are based on mapping the set of all possible motion vectors within the search window in the motion estimation process, into a subset of the original motion vector set. The process of finding the optimal subset of motion vectors is called motion vector quantization

(MVQ). In this method the subset of motion vectors is called a codebook and the motion vectors selected as the element of the codebook are called codevectors. In a general MVQ algorithm first the motion estimation process is implemented using the original motion vector set; therefore, all the blocks within the search window are checked to find the best matching block in the reference frame for the block in the current frame. Based on the statistical characteristics of these motion vectors an initial codebook is defined. For instance the motion vectors with higher frequencies are selected as the codevectors. At the next step each motion vector found from the motion estimation process is approximated by a codevector in the codebook which results in minimum average distortion between the block in the current frame and the block in the reference frame with the reference of the codevector. In other words, each motion vector from motion estimation process is quantized into one of the codevectors in the initial codebook. Each set of motion vectors that are quantized into each codevector called a motion vector cluster. In each cluster the codevector is updated such that the new codevector results in lower distortion. This whole process is repeated until the distortion converges. In the method presented by Lee [14], updating the codebook is implemented only at the encoder side and from one video sequence to another video sequence the codebook may change. In the method presented by Cruz and Woods ([15], [16]), low complexity adaptive MVQ algorithms involve backward adaptive motion vector quantization (BAMVQ) and forward adaptive motion vector quantization (FAMVQ) respectively, which use a general motion vector codebook which is updated periodically in order to make it better suited for coding future motion data. The adaptation algorithm is based on the assumption that if some of the codevectors of the codebook have low use (Least-Frequently-Used strategy or LFU), or have not been used for a significant period of time (Least-Recently-Used strategy or LRU); they may be replaced by new codevectors with higher potential

usefulness resulting in an expected decrease in average distortion. The BAMVQ algorithm [15] presented results in a 10% and 20% bitrate saving for the test sequences Mobile and Tennis respectively; while for the same video sequences, FAMVQ [16] achieved a 14.83% and 21% bitrate saving for Mobile and Tennis test videos compared with the MPEG-2 standard. The motion vector quantization techniques achieve slightly better overall coding results at the cost of higher encoder complexity. The major problem associated with these methods is that larger codebooks would make the motion estimation too slow and complex while smaller sizes restrict the number of possible estimates decreasing motion estimation effectiveness which results in larger block errors. Therefore, these methods are not generally of great interest in practical video coding applications.

2.1.2 Lossless Motion Vector Pre-Encoding Techniques

In the conventional block-based coding scheme, the motion vector of each block is predicted as a function of one or more of its neighboring motion vectors, and then the difference between the current motion vector (MVcur) and predicted motion vector (PMV), motion vector differences (MVD), is compressed using a variable-length coder (VLC) as illustrated in Figure 2.1.

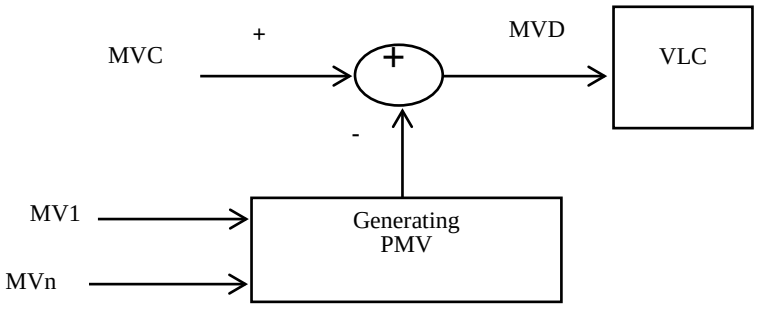


Figure 2.1: Motion vector pre-encoding scheme.

The candidate neighboring motion vectors are spatial and/or temporal neighboring motion vectors. The efficiency of these methods strongly depends upon the accuracy of the PMV.

However, selecting an optimal PMV requires such extensive searches that the increased complexity cannot be ignored. On the other hand, the correlation between neighboring motion vectors is directly related to the motion type in the video sequence; for the video sequences containing complex motions or scene changes the PMV cannot be a proper solution for reducing the motion data since the spatial and temporal correlations between neighboring motion vectors are low for these types of motions respectively. Therefore, the selected PMV is not necessarily close enough to the true motion vector to reduce the motion data. In the following subsections the three main approaches of lossless pre-encoding techniques are described and evaluated in terms of their coding efficiency.

2.1.2.1 Spatial Motion Data Prediction

The correlation between motion vectors corresponding to spatial neighbouring blocks is utilized to generate a spatial predictive motion vector. The main idea of predicting a motion vector of a block from the motion vectors of its neighbouring blocks in the same frame is that it is assumed these blocks belong to a single object in the frame therefore they produce similar motion vectors, but this assumption alleviates the efficiency of this method since a typical video sequence may contain different camera motions and scene changes. Also, in the case where objects are moving along the video sequence, they may have complex motions that mean neighbouring blocks are not necessarily moving with the same velocities. However, besides the formerly mentioned deficiency, spatial PMV has been accepted by the video coding standards because the method provides a balance between the computational simplicity and its capability in reducing the number of bits needed to encode motion vectors better than temporal PMV and spatio-temporal PMV which are described in two following sections respectively. MPEG-1 and MPEG-2 use

first-order intra-frame differential coding of motion vectors to compress motion data ([17], [18]). In H.264/MPEG-4 AVC [19] the PMV of the current block is the median of the MVs of neighbouring blocks in the same frame. The PMV of the current block, MVcur as defined in Equation 2.1, is the median of the three available candidate MVs of neighbouring blocks, which are blocks to the left (A), above (B) and up-right (C) respectively as shown in Figure 2.2.

$$PMV_{cur} = MVA - MVB - MVC + \min(MVA, \min(MVB, MVC)) + \max(MVA, \max(MVB, MVC)) \quad (2.1)$$

The up-left (D) is used when the block C is not available (e.g. if it is outside the current slice), if other blocks shown in Figure 2.2 are not available, the choice of PMV is modified accordingly, and that is described in Section 2.2. The median prediction is very simple and accurate enough for sequences with a relatively uniform motion therefore, it is also obvious that the median PMV is not the optimal PMV in all cases. To improve the predictive motion vector, Kim and Ra [21] proposed to select the motion vector of the block among the same neighbouring blocks as H.264 which results in a minimum MVD. To determine which neighboring motion vector is selected as the predicted one, the mode information (MODE) has to be prepared and sent to the decoder. This method achieved up to a 9.9% motion vector bitrate reduction in comparison with H.264. To improve the selection of spatial the PMV further, Yong et al [20] proposed an encoder that estimates the optimal PMV among a wider set of neighbouring motion vectors (Equation 2.2) without additional information for indicating which predictor is to be used at the encoder side.

$$Motion\ Vector\ Set = \{(MVAx,MVAy), (MVAx,MVBy),\ldots,(MVCx,MVBy),(MVCx,MVBy))\} \quad (2.2)$$

The proposed method decreases the number of bits compared with the H.264/AVC standard by about 2.97% on average. However, as the bitrate improves the number of searches to find the optimal PMV increases and also the bitrate reduction decreases.

2.1.2.2 Temporal Predictive Motion

The predictive motion vector defined in this method is exploited from the motion vectors of temporal neighbouring blocks of the current block which can be the motion vector of block in the previous frame with the same spatial location, called the collocated block, or a function of several temporal neighbouring motion vectors. Using these methods achieved up to 19.09% [22] saving in the motion vector bitrate in comparison with the MPEG-2 standard; however, according to the results presented by Yeh et al [22] there are constraints that limit its efficiency to only specific types of video sequences which contain a moving object with a slow and simple motion along the video sequence. To show the limit of using temporal PMV this work is described as follows: first, it is assumed that the objects in the scene are moving with a constant velocity, e.g. the rectangular object in the first row of Figure 2.2, therefore the motion vector field in the each frame, V [n], can be found from the motion vector field of the previous frame V [n-1]. With this assumption, the motion vector field at the (z + k) location in the n^{th} frame is found, through an operation called autocompensation, from the motion vector field at the location (z) in the previous frame (n-1):

$$V'[n](z + k) = V[n\text{-}1](Z) \tag{2.4}$$

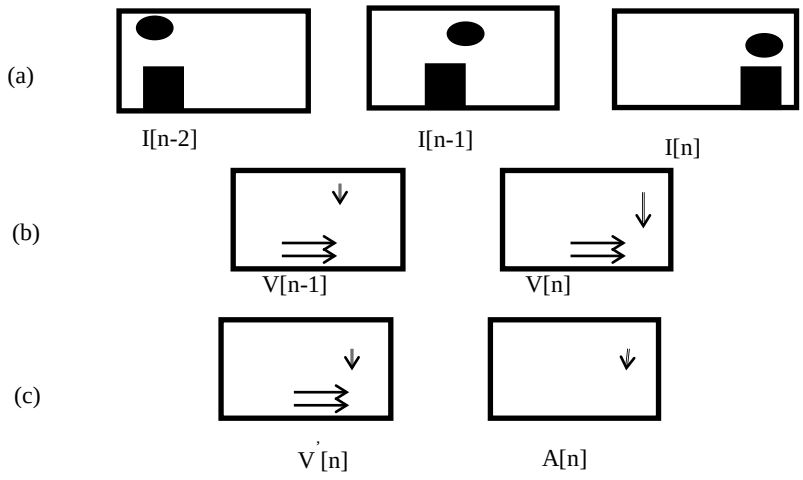
Where

$$k = V[n](z) \tag{2.5}$$

Therefore, the knowledge of V[0] (the first motion vector field) is sufficient to generate the entire sequence. However, in most cases, the velocity of objects is not constant and the error between the compensated motion field V'[n] and the true motion field V[n] is computed as MVD and encoded:

$$A[n] = V[n] - V'[n] \quad (2.6)$$

The energy of A[n] is not necessarily smaller than the energy of V[n], because the acceleration can be due to the change in the direction and not the magnitude of the motion vector field, hence encoding the error is not optimal for all video sequences with complex motions. On the other hand, further refinement of V'[n] in Equation (2.4) in order to reduce the energy of A[n] leads to a decrease in the bitrate at the cost of high complexity which is not desirable for online applications. The results show this method resulted in a 19.09% bitrate saving for encoding motion vectors of the upper left 256x256 quadrant of a 512x512 video sequence which contains a camera panning and zooming on a stationary photograph, and represented simple motion and also the lower right 256x256 quadrant of the same video as a test video sequence with complex motion which led to a 0% improvement in bitrate saving for motion vector encoding.



(a) Hypothetical images of a video sequence (b) Extracted motion sequences (c) Compensated field V' [3] and its difference from V

Figure 2. 3: Temporal predictive motion.

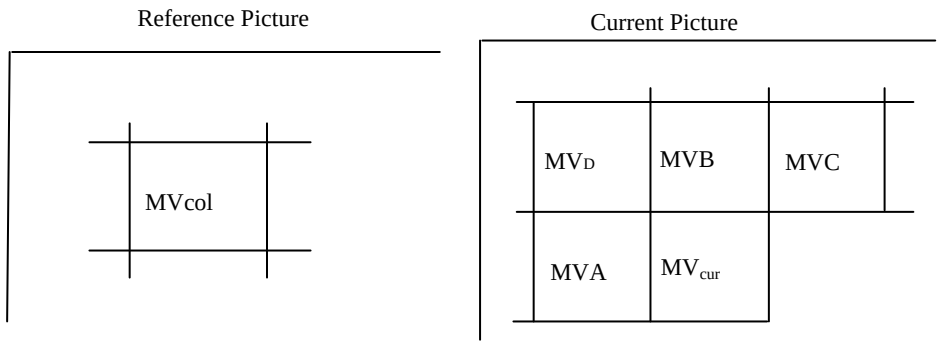


Figure 2.4: Candidate PMV Set.

2.1.2.3 Spatio-Temporal Motion Vector Predictive

Where the simple spatial or temporal coding techniques fail to provide the optimum PMV candidate, there are methods taking advantage from the use of the spatial and temporal redundancies in the motion vector fields. In several works [23-26] the best PMV is chosen out of a set of spatially and temporally selected PMV candidates. However, the extra data transmitted to the decoder to indicate which PMV is selected by encoder decreases the efficiency of the methods with a larger set of candidate PMV. To avoid the transmission of extra signalling data for PMV data recognition at the decoder, the methods proposed by Yang et al [27], select a predictive motion vector which can be automatically selected by the decoder among the candidate predictive motion vectors. To obtain a more precise PMV, by the proposed scheme, both encoder and decoder organize the same candidate set (CS) which consists of all possible distinct candidate PMVs for a current block. As shown in Figure 2.3 CS which is composed of three spatially neighboring MVs, MVA, MVB, MVC and, one temporally neighbouring MV MVcol, and the median MV of the H.264/AVC standard PMV is defined as:

$$CS=\{MVA, MVB, MBC, MVcol, PMV\} \quad (2.7)$$

For each motion position i within the search range, a PMV can be selected automatically by the decoder and a template matching error function T is applied which is given by:

$$PMVi = \arg\min\ T(PMV \mid MVDi) \tag{2.8}$$

where PMV_i is a selected PMV for a tested search position i and its MVD_i. By using the template matching error function, all candidate PMVs in the CS are tested with known information of MVD_i. For each candidate PMV, the template matching error function T calculates the sum of absolute differences (SAD) of neighbouring pixels included in the template matching set (TMS) between the current and reference blocks. In the calculation of the template matching error, a position of the reference block is obtained by a candidate MV, PMV + MVD_i. Finally, a PMV having the minimum template matching error is selected as a decoder selectable PMV for the tested search position. Applying this method for encoding motion vectors contributes a bitrate saving of about 3.21% on average compared to the H.264 standard.

2.1.3 Applied Motion Vector Encoding Techniques in H.264/MPEG-4 AVC

As mentioned previously, a PMV is formed based on previously calculated motion vectors and the MVDs, the difference between the current vector and the predicted vector, is encoded and transmitted. The method of forming a PMV depends upon the motion compensation partition size and on the availability of nearby vectors and macroblock type which can be a macroblock with a motion vector from the motion estimation process or with the SKIP mode in predictive (P) slices and the two DIRECT modes in bi-predictive (B) slices.

2.1.3.1 Predictive Motion Vectors with Available Motion Vector from Motion Estimation Process

Let E be the current macroblock, macroblock partition, or sub-macroblock partition, let A be the partition or sub-partition immediately to the left of E, let B be the partition or sub-partition immediately above E and let C be the partition or sub-macroblock partition above and to the right of E; as shown in Figure 2.4. If there is more than one partition immediately to the left of E, the topmost of these partitions is chosen as A. If there is more than one partition immediately above E, the leftmost of these is chosen as B. Figure 2.4 illustrates the choice of neighboring partitions when all the partitions have the same size (e.g. 16×16) in this case the PMV is computed as below:

1. For the leftmost blocks, PMV is 0.
2. For the rightmost blocks, PMV is computed from modified Equation 2.1 that MVC is replaced by MVD.
3. For the topmost blocks, but not leftmost, PMV is equal to MVA.
4. For other blocks within the frame the PMV is computed from Equation 2.1.

Figure 2.4 shows an example of the choice of prediction partitions when the neighboring partitions have different sizes from the current partition E.

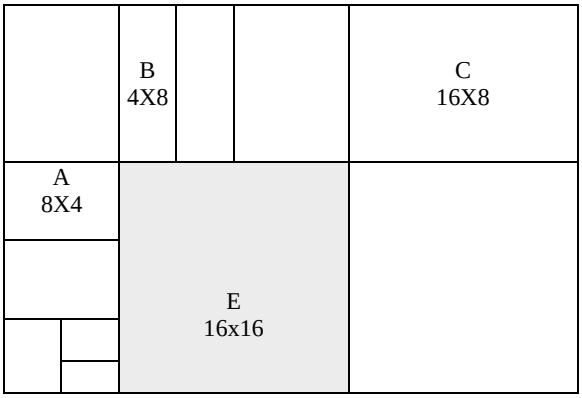


Figure 2.2: Variable block size macroblock partitioning.

1. For transmitted partitions excluding 16×8 and 8×16 partition sizes, PMV is the median of the motion vectors for partitions A, B and C, as shown in Equation 2.1.

2. For 16×8 partitions, the PMV for the upper 16×8 partition is predicted from B and the PMV for the lower 16×8 partition is predicted from A.

3. For 8×16 partitions, the PMV for the left 8×16 partition is predicted from A and the PMV for the right 8×16 partition is predicted from C.

If one or more of the previously transmitted blocks, shown in Figure 2.4, is not available (e.g. if it is outside the current slice), the choice of PMV is modified accordingly. At the decoder, the predicted vector PMV is formed in the same way and added to the decoded MVD.

2.1.3.2 Predictive Motion Vector for Skip and Direct Mode Macroblocks

These modes, when signaled, could in effect represent the motion of a macroblock (MB) or block without having to transmit any additional motion information required by other inter-MB types. Motion for these modes is obtained by exploiting either spatial or temporal correlation of the motion of adjacent MBs or pictures. In the case of skipped macroblocks in P slices, in Extended, Main and Baseline profiles, there is no decoded MVD and a motion-compensated

macroblock is produced using PMV as the motion vector which is generated as the median of three spatial neighbouring blocks, case (1) above for fixed size blocks and variable size blocks respectively.

Skipped macroblocks existing in B slices, in Extended and Main profiles, are predicted using Direct Mode macroblocks or sub-macroblock partitions, either fixed size partitions (blocks) or variable size partitions. No motion vector is transmitted for a B slice macroblock or macroblock partitions encoded in Direct Mode. Instead, the decoder calculates list 0 and list 1 vectors based on previously-coded vectors and uses these to carry out bi-predictive motion compensation of the decoded residual samples. A flag in the slice header indicates whether a spatial or temporal method will be used to calculate the vectors for Direct Mode macroblocks or partitions. In spatial Direct Mode, list 0 and list 1 predicted motion vectors are calculated using the process described in Section 2.1.4.1 If the collocated MB or partition in the first list 1 reference picture has a motion vector that is less than ±1/2 luma samples in magnitude (and in some other cases), one or both of the predicted vectors are set to zero; otherwise the predicted list 0 and list 1 vectors are used to carry out bi-predictive motion compensation. In temporal Direct Mode, the decoder carries out the following steps (Figure 2.5):

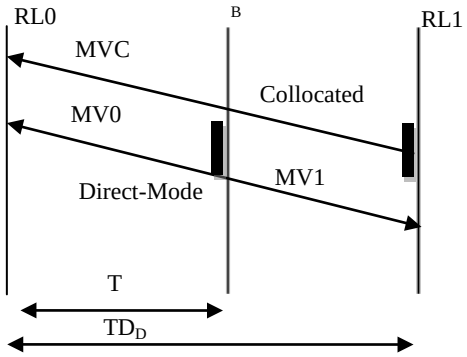


Figure 2.3: Direct-Mode block motion vector estimation.

1. Find the list 0 reference picture for the collocated MB or partition in the list 1 picture. This list 0 reference becomes the list 0 reference of the current MB or partition.

2. Find the list 0 vector, MV, for the collocated MB or partition in the list 1 picture.

3. Scale vector MV based on the picture order count 'distance' between the current and list pictures: this is the new list 1 vector MV1, Equation 2.9.

$$MV1 = \frac{TD_B}{TD_D} \times MV_C \tag{2.9}$$

$$MV2 = \frac{TD_B - TD_D}{TD_D} \times MV_C \tag{2.10}$$

4. Scale vector MV based on the picture order count distance between the current and list 0 pictures: this is the new list 0 vector MV2, Equation 2.10.

These modes are modified when, for example, the prediction reference macroblocks or partitions are not available or are intra coded.

2.2 Motion Vector Entropy Encoding Methods

MPEG-4 uses a predefined Huffman-based table to encode pre-coded MVDs. Exponential Golomb codes [28] (Exp-Golomb) fall into this category. In the H.264 standard, when the entropy encoding mode is set to 0 (Baseline profile) all variable-length coded units except residual block data are coded using Exp-Golomb codes. When the picture parameter flag is set entropy coding mode is set to 1 (Main profile), an arithmetic coding system is used to encode and decode H.264 syntax elements. In this encoding system, Context-based Adaptive Binary Arithmetic Coding (CABAC) [29] achieves good compression performance through (a) selecting probability models for each syntax element according to the element's context, (b) adapting

probability estimates based on local statistics, and (c) using arithmetic coding rather than variable-length coding.

Chapter 3

Encoding of Motion Vectors Using Splines

In this chapter the proposed motion vector encoding scheme is structured as a 3-step algorithm in order to give an overview of the entire process then it will be explained in more detail in Chapters 4 and 5. As explained in Chapter 2, predictive coding schemes that utilize the redundancy between the temporal and spatial neighboring blocks' motion vectors minimize the motion vector data which results in a reduction in the bitrate. In this work a new predictive motion vector (PMV) concept is introduced. The new PMVs are the keypoints[4] of the curves that are fitted to motion vector sets. The concept behind fitting motion vectors to a curve is that as long as a curve is relatively smooth, data can be represented by a set of keypoints (that are used to determine the curve itself) in a much more compact way than labeling each discrete point. As a great deal of motion in video is from either objects moving across the screen, or cameras moving in a controlled manner – smooth curves are easy to come by if we were to map the motion vectors to a curve. There are different curves that can be used, depending on the shape and the control required. Bezier curves [30] are able to produce smooth curves and are used for a

[4] In this work keypoint and control point are used interchangeably.

wide range of applications from animation to character sets. More generalized NURBS curves [31] permit even sharp corners if required and are used for smooth surfaces for modeling or computer aided design. We have selected to use the Hermite spline [32] for its computational and also compression efficiency over previously named curve fitting schemes; Hermite splines use two keypoints and two tangent values at the keypoints for forming its polynomial. In terms of computation, the keypoints sit on the curve; therefore, they can be selected from the original data set which means they can be easily chosen as the data point set's critical points such as extrema values and efficiently being fitted to the data points between each two consecutive keypoints. Bezier curves use four different control points where only two of them sit on the curve and can be selected from the original data point set and the other two control points should be computed such that better curve fitting is achieved. On the other hand, based on the NURBS polynomials [31] a high level of computation is required to reconstruct a data point from the NURBS function. In terms of compression the tangent values needed to characterized the Hermite spline can be automatically computed from the consecutive keypoints which means only two data points, which are keypoints, are sufficient for presenting the Hermite polynomial while Bezier curves require four independent control points and to construct a NURBS keypoints, keypoint weights, and also information about the choice of B-Spline basic function are required in the NURB equation. The choice of tangents is non-unique and there are several options available such that they lead to different types of Hermite spline; Finite Difference Spline [33], Cardinal Spline [34], and Catmull-Rom Spline [35] are three variations of the Hermite Spline. In this work the Catmull-Rom Spline is chosen as the computation for the tangent is simple and efficient enough for motion vector interpolation. By using a curve, any discrete y-axis data (the motion vector) can be extracted at any point by providing the frame number in the x-axis. The

proposed PMV production algorithm contains three stages (Figure 3.1 shows the algorithm of the proposed method);

1. Selecting up to 16 candidate MVs, for each block in each frame in the proposed adaptive motion estimation (AME) process. The number of MV candidates for each block depends on two factors: (1) the pre-determined energy threshold which determines if each block should be encoded in inter or intra mode, to encode a block in inter mode there must exist at least one matching block within the search window of an energy value less than the energy threshold, and (2) the computed bitrate threshold based on which the set of matching blocks selected from previous stage is refined such that only matching blocks of an estimated bitrate less than the bitrate threshold remain as the curve candidates.
2. Choosing the optimum MV from the MV candidates in each set such that the final MV set needs the minimum number of keypoints to be represented by a piecewise Hermite spline curve.
3. Determining keypoints and the header information which are required to reconstruct the MVs of each data point set from its curve at the decoder.

3.1 Determining Motion Vectors through Adaptive Motion Estimation Process

In order to improve the efficiency of the curve fitting process and reduce the encoded data, first for each block up to 16 matching blocks are found depending on the availably of the blocks which meet an energy threshold and bitrate threshold as explained in the previous section. At the first step it is determined that each block is to be encoded in the intra or inter mode. To do so based on an initial threshold, if there exists at least one block of the residual energy less than this threshold the block is encoded in the inter mode, then all the blocks of an energy value less than the threshold within the search window are selected to go through the rate control process to

meet the second constraint (finding a matching block candidate is implemented via a quarter-pixel accuracy motion estimation system). Based on a rate control model, the number of bits for encoding the residual block is predicted using the residual block energy and applied quantization parameter. Therefore, if the number of predicted bits is less than the bitrate threshold the block is selected as one of the matching block candidates from the previously selected blocks in the search window. The threshold value is different for each block and is chosen as the number of bits required for encoding the best matching block which is chosen as the one with the minimum energy within the search window. For a block located at (x,y), where x and y are the horizontal and vertical block references in the f^{th} frame, a set of matching blocks with corresponding set of motion vectors is found as follows:

$$MV_{x,y,f} = \left\{ mv_{x,y,1}, mv_{x,y,2}, ..., mv_{x,y,k} \right\} \qquad for \begin{Bmatrix} 1 \le x \le W \ \ and \ 1 \le \mathrm{y} \le H \\ 1 \le \mathrm{k} \le 16 \ \ and \ 2 \le f \le F \end{Bmatrix} \qquad (3.1)$$

where H and W are the vertical block count and horizontal block count respectively, and F is the number of frames in the video sequence. The frame index, f, is set to be equal to the frame number 2 to the last frame, since the first frame, f = 1, in each set of frame is encoded in the intra mode. Finding the candidate motion vectors for each block, the resulting MVs of the collocated blocks in all the video frames are grouped together to form a set of data points for the curve fitting process (i.e. data points are the motion vectors of each block). For better illustrating the initial data point set each set is presented as a matrix in which the row number is equal to the frame number in the video sequence that block(x,y) is located in it. The number of columns in each row is showing the number of motion vector candidates found in the motion estimation for block(x,y) in that row; therefore, the initial motion vector (IMV), set for block(x,y) is:

$$IMV_{x,y} = \begin{bmatrix} mv_{x,y,1,2} & mv_{x,y,2,2} & .. & mv_{x,y,k_2,2} \\ mv_{x,y,1,3} & mv_{x,y,2,3} & .. & mv_{x,y,k_3,3} \\ : & : & : & : \\ mv_{x,y,1,f} & mv_{x,y,2,f} & .. & mv_{x,y,k_f,f} \\ : & : & : & : \\ mv_{x,y,1,F} & mv_{x,y,2,F} & .. & mv_{x,y,k_F,F} \end{bmatrix} for \begin{Bmatrix} 1 \le k_f \le 16 \\ 2 \le f \le F \end{Bmatrix} \quad (3.2)$$

where k_f is the number of matching blocks found for block(x,y) located in frame f. Note that the number of motion vectors in each row is different, it is chosen to be at most up to 16 matching blocks, since it depends on the number of matching blocks found for each block in the motion estimation process based on the number of bits needed for encoding each block's residual, that is gained from the rate control model and the threshold value. The threshold value is the number of bits needed for encoding the best matching block, the matching block of minimum MSE, as explained previously.

Once the initial motion vector set has been formed, the optimum motion vector candidate should be selected in each row for the curve fitting process. The selected motion vector in row f, or $mv_{x,y,i,f}$ where $1 \le i \le k_f$, is the motion vector of the matching block for block(x,y) in frame f, which is used for motion compensation to reconstruct the block at the decoder.

3.2 Optimum Motion Vector Selection

3.2.1 Ordering MV Candidates

The MV candidates for each block are ordered in ascending order based on the encoded size of each residual block. For simplicity, the number of bits required to encode each residual block is estimated from the same rate control model used in Section 3.1 for selecting the candidate matching blocks in the motion estimation process. Therefore, the motion vectors in each row of

$IMV_{x,y}$ are sorted based on their corresponding residual blocks size, in terms of the number of bits needed to encoded each residual.

$$IMV'_{x,y} = \begin{bmatrix} mv'_{x,y,1,2} & mv'_{x,y,2,2} & .. & mv'_{x,y,k_2,2} \\ mv'_{x,y,1,3} & mv'_{x,y,2,3} & .. & mv'_{x,y,k_3,3} \\ : & : & : & : \\ mv'_{x,y,1,f} & mv'_{x,y,2,f} & .. & mv'_{x,y,k_f,f} \\ : & : & : & : \\ mv'_{x,y,1,F} & mv'_{x,y,2,F} & .. & mv'_{x,y,k_F,F} \end{bmatrix} for \begin{Bmatrix} 1 \le k_f \le 16 \\ 2 \le f \le F \end{Bmatrix} \quad (3.3)$$

Therefore, the residual block bitrate (RBR) of each motion vector, as shown in Equation (3.4), would be true for the motion vectors in each row in matrix $IMV'_{x,y}$:

$$RBR(mv'_{x,y,i,f}) \le RBR(mv'_{x,y,j,f}) \quad for\{i < j \mid 0 \le i, j \le 16\} \quad (3.4)$$

3.2.2 Optimum MV Selection

At this stage, matrix $IMV'_{x,y}$ is transformed into a final motion vector matrix ($FMV_{x,y}$) which has only one column and F rows. $FMV_{x,y}$ is in fact the final set of motion vectors of block position (x,y) in the video sequence frames that are fitted into a Hermite curve and then the curve is represented by its control points which are used to reconstruct the other motion vectors in $FMV_{x,y}$. Therefore, for perfect data compression $FMV_{x,y}$ should be chosen such that the minimum number of control points for the curve is achieved. According to the Hermite spline curve fitting algorithm the control points are the critical points in each data point set, data points are motion vectors in this thesis. Critical points can be the local extrema among the data points. Note that motion vectors are 2-dimensional therefore $IMV'_{x,y}$ is 2-dimensional as well and $IMV'_{x,y,H}$ and $IMV'_{x,y,V}$ are the components of $IMV'_{x,y}$ that contain the horizontal and vertical motion vector elements. Also, $FMV_{x,y}$ containing vertical motion vector elements is represented

by $FMV_{x,y,V}$ and $FMV_{x,y}$ containing horizontal motion vector elements is represented by $FMV_{x,y,H}$. When notations $IMV'_{x,y}$ or $FMV_{x,y}$ are used both vertical and horizontal components of the motion vectors are considered identically.

In order to choose the best motion vector among the candidates first an initial trends (IT_H and IT_V) (which can be increasing, unchanged, or decreasing) is determined for each $IMV'_{x,y}$. To do so the first motion vector in the first row of $IMV'_{x,y}$, $IMV'_{x,y}[0,0]$ is selected as the first data point in the $FMV_{x,y}$, $FMV_{x,y}[0,0]$. The first motion vector on each row is the one that results in the minimum number of bits for encoding its residual blocks based on the motion vector ordering process as described in the previous section.

$$FMV_{x,y}[0,0] = IMV'_{x,y}[0,0] \tag{3.5}$$

Figure 3.1 depicts the motion vector candidates of block(x,y) in the first three video frames. The first data point, that corresponds to the block(x,y) in the second frame, is the one with a smaller column number which is the one that results in the minimum number of bits for encoding its residual block and in this example it is equal to (2,2):

$$\begin{aligned} FMV_{x,y,H}[0,0] &= IMV'_{x,y,H}[0,0] = 2 \\ FMV_{x,y,V}[0,0] &= IMV'_{x,y,V}[0,0] = 2 \end{aligned} \tag{3.6}$$

Then in order to determine the initial trends $FMV_{x,y,V}[0,0]$ and $FMV_{x,y,H}[0,0]$ are compared with 0, described as follows:

Case 1: IT_V and IT_H are equal:

- Case 1.1.1[5]: Both are larger than zero, IT_V and IT_H are increasing.
- Case 1.1.2: Both are smaller than zero, IT_V and IT_H are decreasing.

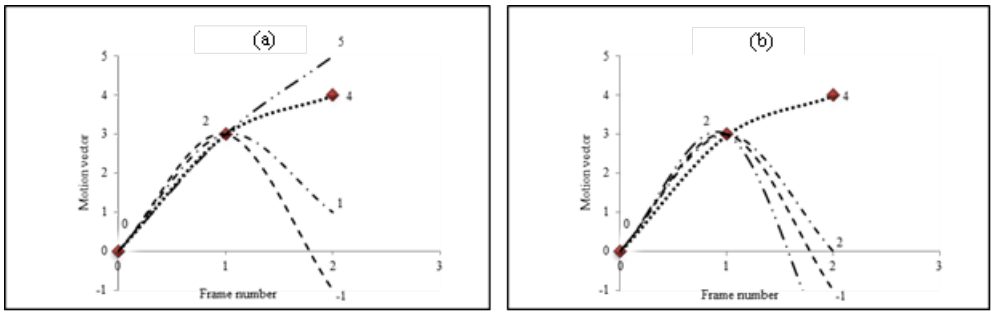
[5] Cases 1.1 and Cases 1.2 correspond to selecting the first data point and the second data point respectively.

- Case 1.1.3: Both are zero, IT_V and IT_H are equal.

To determine the second data point ($FMV_{x,y}[1,0]$) in Case 1, the current trends (CT_H and CT_V) between FMV[0,0] and the motion vectors on the second row of $IMV'_{x,y}$ are computed. Based on IT_H (or IT_V as in Case 1 IT_V and IT_H are equal) four cases may occur for selecting the next data point:

- Case 1.2.1: There exists a motion vector for which CT_H and CT_V are equal to the IT_H (IT_V), therefore motion vector is selected.

  In Figure 3.1, comparing $FMV_{x,y,V}[0,0]$ and $FMV_{x,y,H}[0,0]$ with zero, IT_H and IT_V are both increasing (Case 1.1.1). Therefore, for choosing the second data point corresponding to the block(x,y) in the third frame (the first frame is encoded in inter-mode), if there is a motion vector that CT_H and CT_V are both increasing that motion vector is selected (e.g. for the motion vector candidates in Figure 3.1, the motion vector on the second column of $IMV'_{x,y}$ meets the condition provided in Case 1.2.1).



(a) Horizontal motion vector component (b) vertical motion vector component

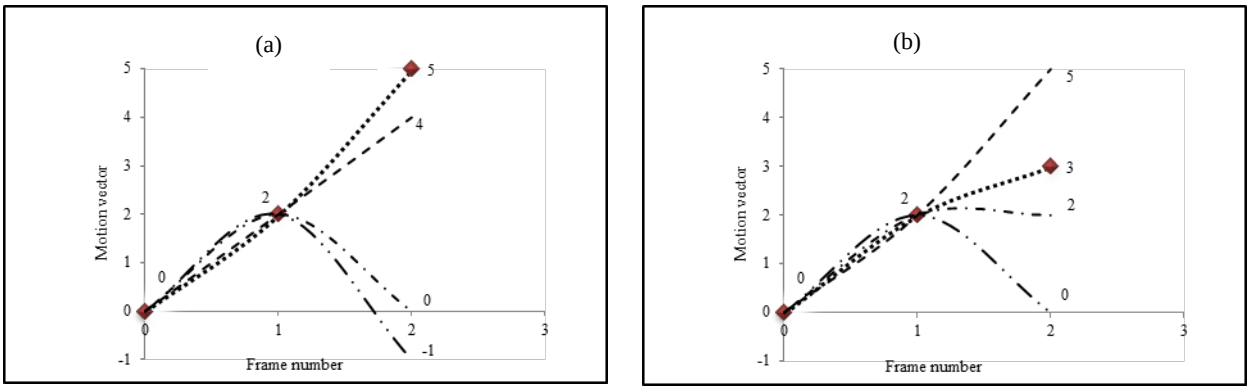
Figure 3.1: Case 1.2.1 optimum motion vector selection for the second data point.

First Column Second Column Third Column Fort Column

$$FMV_{x,y,H}[1,0] = IMV'_{x,y,H}[1,2] = 4$$
$$FMV_{x,y,V}[1,0] = IMV'_{x,y,V}[1,2] = 4 \quad (3.7)$$

- Case 1.2.2: There may be more than one such motion vector as in Case 1.2.1 therefore the motion vector with the least number of bits for encoding its residual block is selected, or in other words, the motion vector with the smaller column number is selected, as in Figure 3.2.

$$FMV_{x,y,H}[1,0] = IMV'_{x,y,H}[1,2] = 5$$
$$FMV_{x,y,V}[1,0] = IMV'_{x,y,V}[1,2] = 3 \quad (3.8)$$



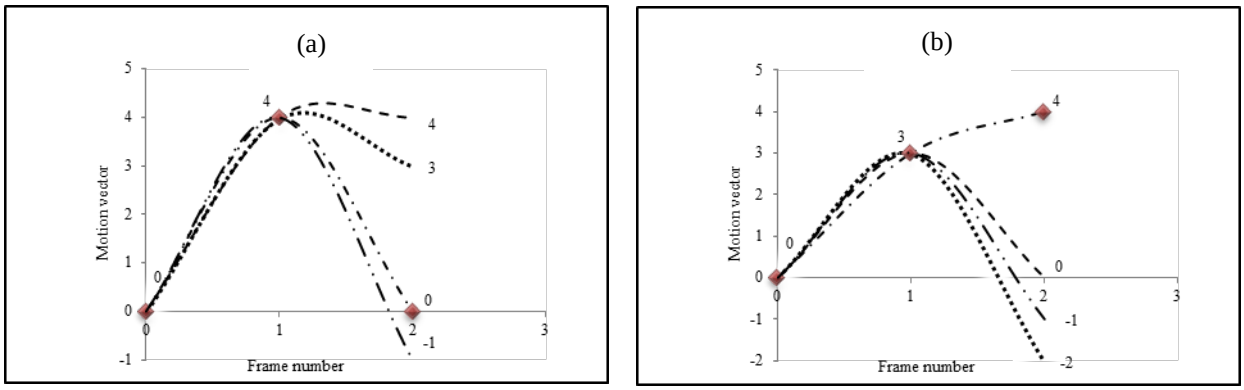
(a) Horizontal motion vector component (b) vertical motion vector component

Figure 3.2: Case1.2.2 optimum motion vector selection for the second data point.

First Column Second Column Third Column Fort Column

- Case 1.2.3: There is a motion vector for which only CT_H (CT_V) is equal to IT_H (IT_V) and there is no motion vector matching any of the above cases, in this situation this motion vector is selected. If there is more than one such a motion vector then the one with the lower column number is selected. As in the example presented in Figure 3.3, among the motion vectors on the second column only CT_V is equal to the IT_V and there is no motion

vector matching Cases 1.2.1 and 1.2.2. In this case the motion vector on the second column is selected.



(a) Horizontal motion vector component (b) vertical motion vector component

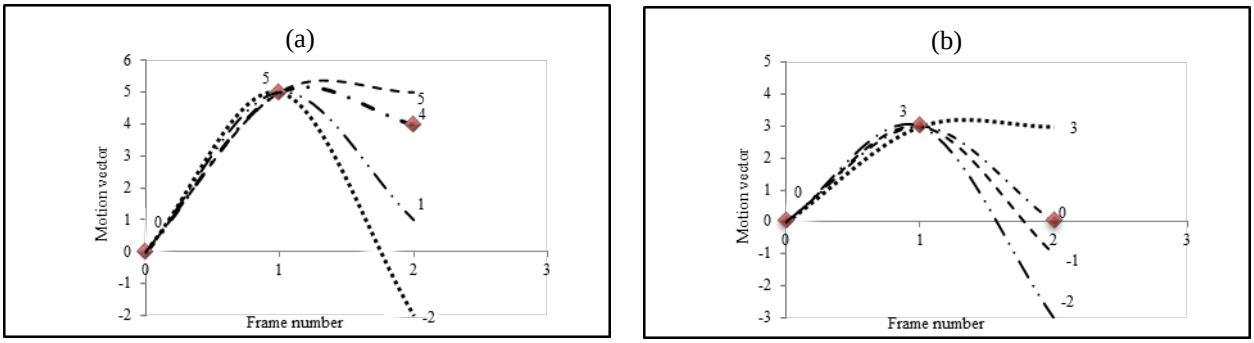
Figure 3.3: Case 1.2.3 optimum motion vector selection for the second data point.

First Column Second Column Third Column Fort Column

$$FMV_{x,y,H}[1,0] = IMV'_{x,y,H}[1,2] = 0$$
$$FMV_{x,y,V}[1,0] = IMV'_{x,y,V}[1,2] = 4 \quad (3.9)$$

- Case 1.2.4: There is no motion vector that matches any of the above conditions, therefore the first motion vector on the row is selected since in this situation all the motion vectors on the row are equal in terms of their property for the curve fitting, they do not affect the number of control points, since to maximize the compression efficiency the motion vector with minimum number of bits for encoding its residual block is selected, as shown in Figure 3.4.

$$FMV_{x,y,H}[1,0] = IMV'_{x,y,H}[1,2] = 4$$
$$FMV_{x,y,V}[1,0] = IMV'_{x,y,V}[1,2] = 0 \quad (3.10)$$



(a) Horizontal motion vector component (b) vertical motion vector component

Figure 3.4: Case 1.2.4 optimum motion vector selection for the second data point.

First Column Second Column Third Column Fort Column

Case 2: IT_V and IT_H are different:

- Case 2.1.1: IT_V is larger than zero, IT_V is increasing.
  - Case 2.1.1.1: IT_H is smaller than zero, IT_H is decreasing.
  - Case 2.1.1.2: IT_H is zero, IT_H will be equal.
- Case 2.1.2: IT_V is smaller than zero, IT_V is decreasing.
  - Case 2.1.2.1: IT_H is larger than zero, IT_H is increasing.
  - Case 2.1.2.2: IT_H is zero, IT_H will be equal.
- Case 2.1.3: IT_V is zero, IT_V will be equal.
  - Case 2.1.3.1: IT_H is larger than zero, IT_H is increasing.
  - Case 2.1.3.2: IT_H is smaller than zero, IT_H is decreasing.

To determine the second data point ($FMV_{x,y}[1,0]$) in Case 2, the current trends (CT_H and CT_V) between FMV[0,0] and the motion vectors on the second row of $IMV'_{x,y}$ are computed. Based on IT_H and IT_V following cases may occur for selecting the next data point:

- Case 2.2.1: There exists a motion vector for which CT_H and CT_V are equal to the IT_H and IT_V respectively, therefore this motion vector is selected.

- Case 2.2.2: There may be more than one such a motion vector as in Case 2.2.1 on the second row of $IMV'_{x,y}$, the motion vector with the least number of bits for encoding its residual block is selected (the motion vector with smaller column number is selected).
- Case 2.2.3: There is a motion vector for which only CT_H is equal to IT_H and there is no motion vector as in Case 2.2.1. The algorithm checks to see if there is a motion vector that its corresponding CT_V is equal to IT_V, if there is such a motion vector, between the motion vector that has met the condition: $CT_H = IT_H$ and motion vector that has met the condition: $CT_V = IT_V$, the one with the smaller column number is selected.
- Case 2.2.4: There is no motion vector that matches none of the above conditions, therefore the first motion vector on the row is selected since in this situation all the motion vectors on the row are equal in terms of their property for the curve fitting, they do not affect the number control points, since to maximize the compression efficiency the motion vector with minimum number of bits for encoding its residual block is selected.

At the end of this process matrix FMV is formed such that its motion vectors (the curve data points) are the best selected points for the curve fitting process and their residual block compression. In order to choose the next data points among the motion vector candidates on the other rows of $IMV'_{x,y}$, the same algorithm is implemented but the IT_H and IT_V for each row are replaced by the trend between the last two data points in $FMV_{x,y}$.

3.3 Curve Fitting Algorithm

Curve fitting is the process of constructing a curve, or mathematical function, which has the best fit to a series of data points. In other words, the curve control points (or the coefficients of its mathematical function) are sufficient to reconstruct the data points fitted into the curve from the curve polynomial. Therefore, if proper data points and also proper curve type are chosen there is

a high probability that the number of control points is less than the number of data points, using a smaller set of control points to represent the data points results in data compression. Selecting proper data points is done in the proposed adaptive motion estimation process (Section 3.2.2) in which motion vectors are determined such that the final motion vector sets ($FMV_{x,y,V}$ and $FMV_{x,y,H}$) contain motion vectors that need a minimum number of control points to be reconstructed by a curve. As explained, the Hermite spline is chosen since it is very easy to calculate, but also very powerful and is used to smoothly interpolate between two key points. In the following sections first a typical Hermite curve fitting algorithm is explained, Section 3.3.1, and then in Section 3.3.2 the method of reconstructing the data points (motion vectors) from the control points of the curve is explained.

3.3.1 Cubic Hermite Curve Fitting

A cubic Hermite spline is a third-degree polynomial that consists of four Hermite basis functions, Figure 3.5, each multiplied by a coefficient.

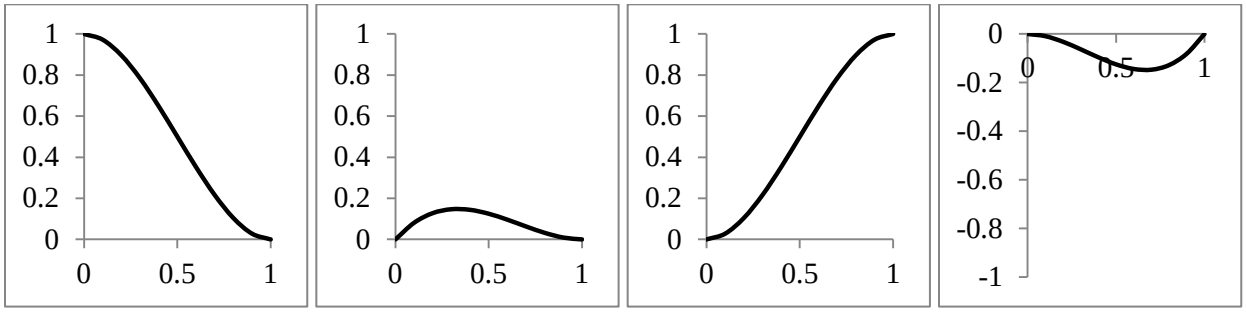


Figure 3.5: Hermite Basis Functions.

The basis functions' coefficients are chosen as a set of two control points and two control tangents. A typical Hermite polynomial is defined as:

$$H(t) = P_0 h_{00} + m_0 h_{10} + P_1 h_{01} + m_1 h_{11} \quad (3.9)$$

$$h_{00} = (2t^3 - 3t^2 + 1) \quad (3.10)$$

$$h_{10} = (t^3 - 2t^2 - t) \quad (3.11)$$

$$h_{01} = (-2t^3 + 3t^2) \quad (3.12)$$

$$h_{11} = (t^3 - t^2) \quad (3.13)$$

where h_{00}, h_{10}, h_{01} and h_{11} are the Hermite basis functions and, p_0 and p_1 are the start and end control points respectively, and m_0 and m_1 are the control tangents defining how the curve enters and leaves a control point (e.g. direction and speed).

As explained in Section 3.2.2, control points are the local maximum and minimum of each data point set in including the end points (first and the last data points). Therefore, if there exist more than two local extrema in a set of data points, more than one Hermite spline (or piece wise Hermite spline) is used for fitting to the data points. For instance, Figure 3.6 depicts the set of data points, {-3, 0, 2, 5, 3, 2, -3, -3, -3, -4, 0}, which contains six local extrema, the crossed points on the figure, containing two end points (-3) and (0) and one maximum (5) and one minimum (-4) and two critical points the first (-3) and the third (-3). To fit a Hermite curve to such a data point set, six curves are used for the subsets {-3, 0, 2, 5}, {5, 3, 2, -3}, {-3, -3, -3}, {-3, -4} and {-4, 0}. As we can see the second control point of each part is the first control point of the next part. In order to calculate the tangents at the control points of each subset as it was explained in the chapter introduction, the choice of tangents is non-unique and there are several options available such that they lead to different types in the Hermite spline family.

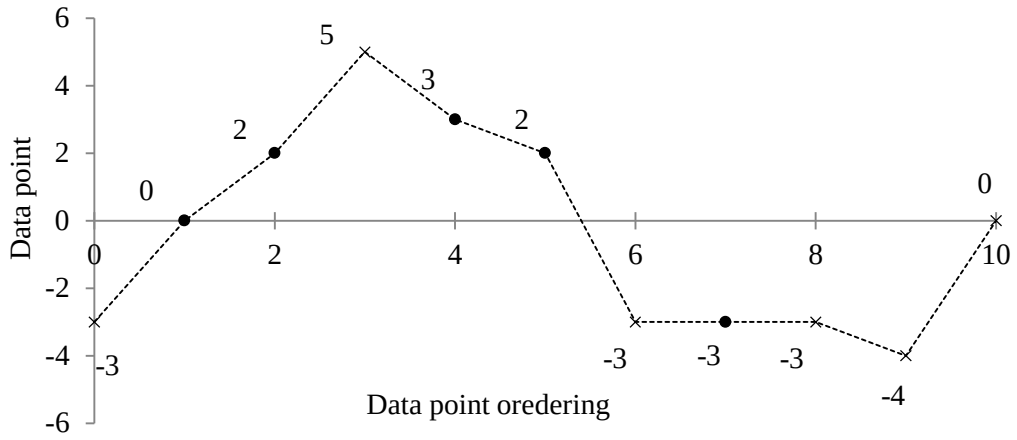


Figure 3.6: Assigning data point set control points.

In this work the Catmull-Rom Spline is chosen as the computation of the tangent as it is simple and also the tangent of each control point is directly computed from its neighbouring control points;

$$m_i = 0.5 \times (p_{i+1} - p_{i-1}) \tag{3.14}$$

where m_i is the tangent at the i^{th} control point (p_i) and p_{i-1} and p_{i+1} are the prior and next control points of p_i. For computing the first control point tangent the equation of the tangents would be:

$$m_i = 0.5 \times (p_{i+1} - p_i) \tag{3.15}$$

and the last control point tangent is equal to:

$$m_i = 0.5 \times (p_i - p_{i-1}) \tag{3.16}$$

Considering the data point set in Figure 3.6, with control point set equal to {-3, 5, -3, -3, 4, 0}, and the tangent set would be {8, 0, -4, 3.5, -1.5, 4}. Therefore, except the control points no extra data is needed for the tangent values for reconstructing the data points from the curve. In order to reconstruct data points in each subset, interval [0, 1] is divided into the number of data points in the subset including the control points, then by assigning each value (t) in the interval to each of the data points in the subset in order, data points are reconstructed using Equation (3.9). For

example, for the first subset of the previous example which contains four data points, the reconstructed data points from the curve are computed by assigning t to 0, 1/3, 2/3 and1 for each data point in the subset respectively. Figure 3.7 illustrates the piecewise Hermite curve consisting of five Hermite splines corresponding to the five subsets of the data point sets in the previous example with Equations (3.17.a-3.17.e):

$$H_1(t) = -3\times(2t^3 - 3t^2 + 1) + 8\times(t^3 - 2t^2 - t) + 5\times(-2t^3 + 3t^2) \quad (3.17.a)$$

$$H_2(t) = 5\times(2t^3 - 3t^2 + 1) - 3\times(-2t^3 + 3t^2) - 4\times(t^3 - t^2) \quad (3.17.b)$$

$$H_3(3) = -3\times(2t^3 - 3t^2 + 1) - 4\times(t^3 - 2t^2 - t) - 3\times(-2t^3 + 3t^2) + 3.5\times(t^3 - t^2) \quad (3.17.c)$$

$$H_4(t) = -3\times(2t^3 - 3t^2 + 1) + 3..5\times(t^3 - 2t^2 - t) - 4\times(-2t^3 + 3t^2) - 1.5\times(t^3 - t^2) \quad (3.17.d)$$

$$H_5(t) = -4\times(2t^3 - 3t^2 + 1) - 1.5\times(t^3 - 2t^2 - t) + 4\times(t^3 - t^2) \quad (3.17.e)$$

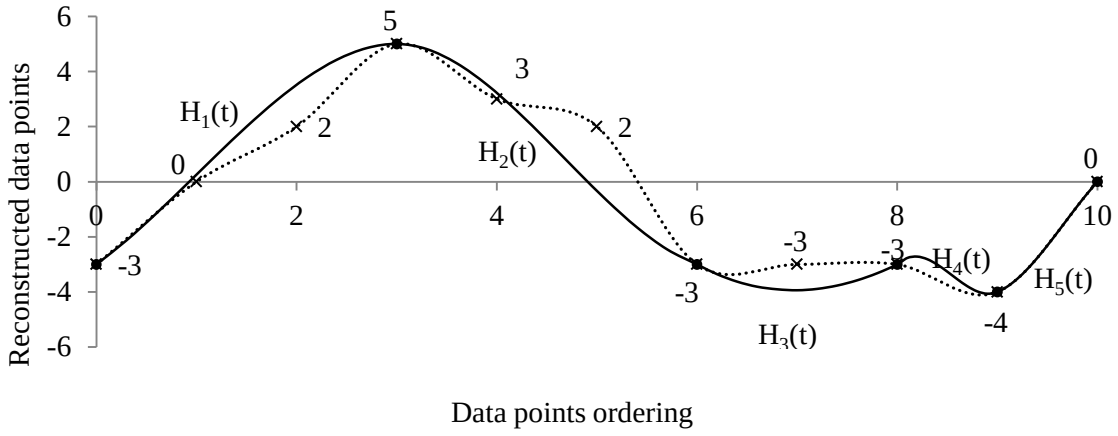


Figure 3.7: Piecewise Hermite spline fitted to the set {-3, 0, 2, 5, 3, 2, -3, -3, -3, -4, 0}.

3.3.2 Appling Hermite Curve Fitting to Motion Vector Encoding

Column matrices $FMV_{x,y,H}$ and $FMV_{x,y,V}$ obtained from the previous step of the algorithm, respectively contain the horizontal and vertical components of the motion vectors of block(x,y) along the video sequence, where (x,y) is the temporal address of the collocated blocks which are the x^{th} horizontal and y^{th} vertical blocks located in the frames of the video sequence. Based on the proposed curve fitting algorithm, while elements on the same row of these matrices belong to

the same matching block, the motion vectors of each $FMV_{x,y,H}$ and $FMV_{x,y,V}$ are treated as two separate data point sets to be fitted into a curve. Therefore, assuming each frame contains M×N blocks (where M and N are the horizontal and vertical number of blocks in the video frame) there would be M×N $FMV_{x,y,V}$ and M×N $FMN_{x,y,H}$ matrices that each matrix is fitted into a separate curve. The same curve fitting algorithm is applied for $FMV_{x,y,H}$ and $FMV_{x,y,V}$ and they are called the data point set in the curve fitting algorithm. Based on the Hermite curve fitting algorithm (Section 3.3.1) the control points of each data point set ($FMN_{x,y,H}$, $FMV_{x,y,V}$) are found, that are the local extrema of motion vectors in each matrix. Since the reconstructed data points from the curve are not necessarily equal to their original values and the proposed motion vector encoding is lossless, all the motion vectors should be recovered at the decoder conforming to their original value. For this purpose, at the encoder side all the data points are reconstructed using the control points and the tangents of each set. Since the original data points are integer, four situations (defined as categories) may happen for each data point. Therefore, each set of data point is mapped into one of these categories:

- Category One: This contains control points.
- Category Two: The non-integer reconstructed data points from the curve that should be rounded to their upper integer at the decoder side, and also the data points, except keypoints, that are recovered directly from the curve (any non-control points with integer reconstructed values from the curve that belong to this category are the real values of the motion vectors).
- Category Three: The non-integer data points reconstructed from the curve that should be rounded to their lower integer at the decoder side.

- Category Four: The motion vectors which the difference between their original value and their reconstructed data point from the curve is more than one, for such points the original data are transmitted.

Table 3.1 shows the results of categorizing the motion vectors of the first 15 frames of QCIF Foreman test video sequence of the 4×4 block size motion estimation and block position (10, 13).

Table 3.1: Categorizing motion vectors based on their reconstructed data points

$FMN_{10,13,H}$	0	NA	NA	1	1	0	-2	-5	-6	-8	0	3	0	-3	-5
Data point	0	-	-	1	1	-0.54	-2.91	-5.40	-7.33	-8	-2.37	3	2.2	0.259	-2.3
Category	One	-	-	One	One	Three	Three	Two	Four	One	Four	One	Four	Four	One
$FMN_{10,13,H}$	0	NA	NA	1	1	1	2	3	3	3	0	2	0	1	2
Data point	0	-	-	1	1	1	2	3	3.31	3	0	2	0	0.87	2
Category	One	-	-	One	Four	One	Two	One	Two	One	One	One	One	Two	One

Categorizing motion vectors leads to reconstructing them at the decoder based on their original values; however, based on the coded video sequence format used in H.264 standard it does not mean that the category index for all the motion vectors should be encoded. In order to describe which category indices should be encoded and contained in the video data bitstream, the coded video sequence format used in H.264 is explained in the next section**.**

3.4 Coded Data Format

The encoded video signal (a sequence of bits representing the coded video data) consists of three main layers, the Video Coding Layer, the Slice Layer, and the Macroblock Layer. While the header and the data units constitute each layer, each of the lower layers is a subset of its upper layers.

3.4.1 Video Coding Layer (VCL)

In order to make use of coding-specific features at the VCL and transport-specific features at network abstraction layer (NAL), in H.264 they are specified separately. A coded video sequence at its upper layer level (VCL) consists of NAL units (Figure 3.8) that are transmitted over a packet-based network, bitstream transmission lines, or stored as a file. Each NAL unit contains an integer number of bytes; the first byte is the header specifying the type of data in the NAL unit and the remaining bytes contain payload data of the type indicated by the header. The sequence of payload data bytes is represented by the Raw Byte Sequence Payload (RBSP). An example of a sequence of RBSP units is shown in Figure 3.9 each RBSP unit is a NAL data unit. Table 3.2 gives brief descriptions about different types of RBSP units.

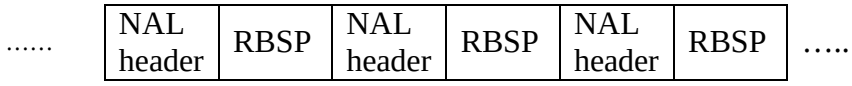


Figure 3.8: Sequence of NAL units.

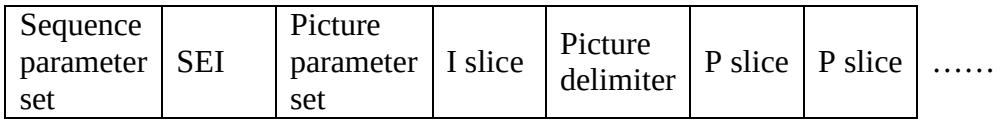


Figure 3.9: Example showing a sequence of RBSP elements

In Figure 3.9 parameters in the sequence parameter set include an identifier (seq-parameter-set-id), limits on frame numbers, picture order count, the number of reference frames that may be used in decoding (including short and long term reference frames), the decoded picture width and height, and the choice of progressive or interlaced coding. While a sequence parameter set contains parameters to be applied to a complete video sequence (a set of consecutive coded pictures), a picture parameter set contains parameters which are applied to one or more decoded pictures within a sequence. A picture parameter set includes an identifier (pic-parameter-set-id),

a selected seq-parameter-set-id, a flag to select VLC or CABAC, the header, and the data units constitute each layer. Therefore, the information stored in the parameter set units remains unchanged in our proposed method as they represent the information about the aspects of the video encoding process such as the type of entropy encoder and the structure of each picture or a sequence of pictures; however, they contain neither the addressing information such as macroblock index, the frame number, or the encoded macroblock values such as residual data and the motion data.

3.4.2 Slice Layer

Each video sequence may be sampled as a series of complete frames or as a sequence of interlaced fields. A field (of interlaced video) or a frame (of progressive or interlaced video) is encoded to produce a coded picture. A coded picture consists of a number of macroblocks, each containing 16×16 luma samples and associated chroma samples (8×8 C_b and 8×8 C_r samples in H.264 standard). Within each picture, macroblocks are arranged into slices, where a slice contains a set of macroblocks in raster scan order (but not necessarily contiguous)[6] and a slice header (Figure 3.10). Depending on the types of macroblocks that exists in each slice there are five slice types; an I slice may contain only I macroblock types (I macroblock type refers to an inter-coded macroblock, covered by the Extended, Main and Baseline profiles in H.264), a P slice may contain P and I macroblock types (P macroblock type refers to an intra-coded macroblock which can be predicted based on preceding frames as its prediction reference frames,

[6] The Baseline Profile supports Arbitrary Slice Order which means that slices in a coded frame may follow any decoding order. ASO is defined to be in use if the first macroblock in any slice in a decoded frame has a smaller macroblock address than the first macroblock in a previously decoded slice in the same picture.

Table 3.2: Brief descriptions about different type of RBSP units

RBSP type	Description
Parameter Set	'Global' parameters for a sequence such as picture dimensions, video format, macroblock allocation map[7]. The parameter set may belong to a picture or a sequence of pictures.
Supplemental enhancement information (SEI)	Side messages that are not essential for correct decoding of the video sequence.
Picture delimiter	The boundary between video pictures (optional). If not present, the decoder infers the boundary based on the frame number contained within each slice header.
Coded slice	Header and data for a slice; this RBSP unit contains actual coded video data.
Data partition A, B or C	Three units containing Data Partitioned slice layer data
End of sequence	Indicates that the next picture (in decoding order) is an IDR[8] picture (Not essential for correct decoding of the sequence).
End of stream	Indicates that there are no further pictures in the bitstream. (Not essential for correct decoding of the sequence).
Filler data	Contains 'dummy' data (may be used to increase the number of bytes in the sequence). (Not essential for correct decoding of the sequence).

also covered by Extended, Main and Baseline profiles in H.264), and a B slice may contain B and I macroblock types (B macroblock type refers to an intra-coded macroblock which can be predicted based on both the preceding and following frames as its prediction reference frames, covered by Extended and Main profiles in H.264). There are two further slice types, SI and SP,

[7] The allocation of macroblocks is determined by a macroblock to slice group map that indicates which slice group each MB belongs to.

[8] An encoder sends an IDR (Instantaneous Decoder Refresh) coded picture (made up of I- or SI-slices) to clear the contents of the reference picture buffer. On receiving an IDR coded picture, the decoder marks all pictures in the reference buffer as 'unused for reference'. All subsequent transmitted slices can be decoded without reference to any frame decoded prior to the IDR picture. The first picture in a coded video sequence is always an IDR picture.

which are supported by the Extended profile in H.264. SP and SI slices are specially-coded slices that enable (among other things) efficient switching between video streams and efficient random access for video decoders [36]. Slices of a video frame may be arranged into one or more slice group(s). Multiple slice groups make it possible to map the sequence of coded MBs to the decoded picture in a number of flexible ways (further details on the applications and the slices to slice group map types are represented may be found [36]).

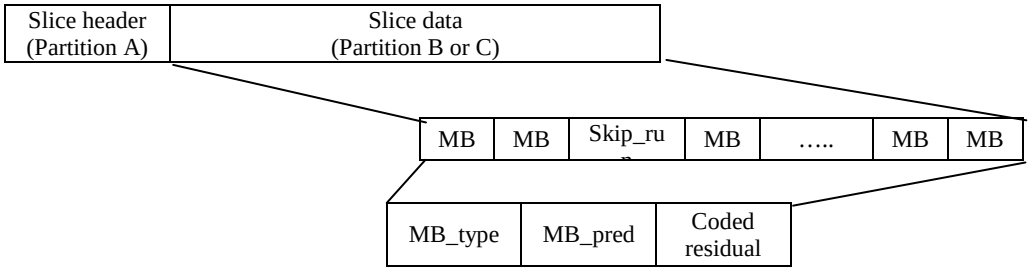


Figure 3.10: Slice syntax.

As a typical encoded video signal layer, the slice layer also contains a header and data units (Figure 3.11) which divide each slice into three partitions: partition A contains the slice header and header data for each macroblock in the slice. The slice header (or partition A) conveys information common to all macroblocks in the slice, such as the slice type which determines which macroblock types are allowed, the frame number that the slice corresponds to, reference picture settings, and default quantization parameter (QP); therefore, if portion A is highly sensitive to the transmission errors since it is lost it would be difficult (or at some points impossible) to decode the related slice. Partition B contains coded residual data for Intra and SI slice macroblocks and coded residual data for inter coded macroblocks (forward and bi-directional) are referred to as Partition C. Partitions B and C can (with careful choice of coding parameters) be made to be independently decodable and so a decoder may (for example) decode A and B only or A and C only, lending flexibility in an error-prone environment.

3.4.3 Macroblock Layer

The macroblock layer contains all the information necessary for decoding the macroblock, Figure 3.2, the header unit mb_type contains the macroblock type, I, P, B, SI or SP, and mode information showing whether or not the macroblock is partitioned into smaller partitions (sub-partitions) and how it is partitioned as in H.264 a macroblock can be partition into variable block size sub-macroblocks. Each macroblock can be divided into two 16×8, two 8×16 or four 8×8 sub-macroblocks and also each 8×8 sub macroblock can be divided into four 4×4 blocks, depending on the macroblock block mode and type signaled in mb_type. Other necessary information such as P and B macroblocks' and sub-macroblocks' motion vector differences (MVDs) and reference frames' indices which are needed to reconstruct the macroblock from its residual data, are contained in mb_pred header unit. The residual data is contained in coded residual units in the macroblock layer (further details are available [34])

Based on the data arrangement in an encoded video bitstream, explained in the previous section, we discuss how the motion information from the proposed method are embedded in the encoded video bit stream and the motion information in the mb_pred unit (motion vector differences (or MVDs)) in the macroblock layer is replaced by the new motion information from proposed method.

Table 3.3: New motion information placement in the bitstream

Motion information placed in mb_pred unit	Motion vector category
Motion vector Control Points	One
No data Encoded	Two
Category Index Flag	Three
Motion Vector with Category Index Flag	Four

The new motion information contains motion vectors which correspond to Category One (control points) and Category Four and also the Category indices (explained in Section 3.3.2) for all the motion vectors. Therefore, only for a subset of blocks there exist motion vectors in each block's header; however, in order to distinguish whether or not the encoded motion vector existing in the mb_pred unit in replacement of MVD is a control point of the corresponding curve or it is just a motion vector of Category Four. The motion vector for one of these two motion vector types the category indices are encoded and placed in mb_pred. For the rest of the block with motion vectors of Categories Two and Three no motion vector is encoded only for distinguishing which category the reconstructed motion vector belongs to, for one these two motion vector types the category indices are encoded and placed in mb_pred unit instead of MVD data as well (as shown in Table 3.1). In order to choose between Categories One and Four, and between Categories Two and Three, extensive searches on different video sequences has been implemented and based on their results it would be more beneficial to encode Category Four and Three in terms of bitrate (more details on the statistics of the results are presented in Chapter 5).

3.5 Entropy Encoding Process

The resulting motion information, control points, and category indices, are Exponential-Golomb (Exp-Golomb) entropy encoded since the proposed method is compared with the Baseline profile in H.264 standard in which all the syntax elements except residual data are encoded based on Exp-Golomb encoding scheme. The residual data is always encoded by the CAVLC encoding technique.

3.5.1 Exponential-Golomb Encoding Technique

Exp-Golomb codes are variable length universal[9] codes with regular constructions. The first 9 codewords for the first 9 positive integers (code_num) are given in Table 3.4 from examining the codewords in Table 3.4 it is clear that there is a logical way in constructing Exp-Golomb codes:

$$[\text{M zeros}][1][\text{Info}] \tag{3.18}$$

INFO is M-bit field carrying information. The first codeword has no leading zero or trailing INFO. The first two codewords have a single-bit INFO field; codewords 3–6 have a two-bit INFO field, and so on. The length of each Exp-Golomb codeword is (2M + 1) bits and each codeword can be constructed by the encoder based on its index code_num:

$$M = \text{floor}\ (\log_2 [\text{code\_num} + 1]) \tag{3.19}$$

$$\text{Info} = \text{code\_num}\ + 1 - 2M \tag{3.20}$$

The algorithm of constructing an Exp-Golomb code contains three steps:

1. Add 1 to the code_num mathematically.
2. Convert the added code_num into it binary representation.
3. Count the number of bits from the step 2, subtract one, and write that number of starting zero bits preceding the previous bit string.

[9] A universal code for integers is a prefix code that maps the positive integers onto binary code words with the additional property that whatever the true probability distribution on integers, as long as the distribution is monotonic (i.e., $p(i) \geq p(i + 1)$ for all positive i), the expected lengths of the code words are within a constant factor of the expected lengths that the optimal code for that probability distribution would have assigned.

Table 3.4: Exp-Golomb codewords

Code_num	Code word
0	1
1	010
2	011
3	00101
4	00110
5	00111
6	0001000
7	0001001
…	…

A codeword can be decoded as follows:

1. Read in M leading zeros followed by 1.
2. Read M-bit INFO field.
3. Code_num = 2M+ INFO – 1.

(For codeword 0, INFO and M are zero.)

Since the Exp-Golomb maps only positive integers into a their binary representation, and motion vectors can be negative integer, in H.264 all the motion vector differences are mapped into positive integers as follow;

1. code num = $2|k|$ $(k \leq 0)$
2. code num = $2|k|-1$ $(k > 0)$

This mapping is designed to produce short codewords for frequently-occurring values and longer codewords for less common parameter values. For example, the commonly-occurring motion vector difference (MVD) value of 0 maps to code_num 0 whereas the less-common MVD= −3 maps to code_num 6. In this work, the same method is used for sign mapping the control points into positive integers (Note: if the motion estimation is of the fractional pel accuracy, as in this

work ¼ pel motion estimation, the integer parts and the non-integer part of the motion vector are considered as two separate integers)

The category indices that should be encoded, the forth and the third categories, are also Exp-Golomb encoded while for saving bitrate the most occurring category indices are mapped to zero and the other is mapped to one as in Table 3.4 the number of bits needed to encode zero and one are one bit and three bits respectively. Therefore, in order to decide which category indices should be encoded by a single bit based on the statistics of the test results on different video sequences proved that category index forth is most rare occurring category among other categories that will be explained in details in Chapter 5.

3.5.2 Context-Based Adaptive Variable Length (CAVLC) Entropy Encoding

The zig-zag ordered 4×4 blocks of quantized transformed coefficients are CAVLC entropy encoded. CAVLC encoding scheme takes advantage of several characteristics of quantised 4×4 blocks; after motion estimation (prediction), quantized transformed blocks typically contain mostly zeros. A run-level encoding technique applied in CAVLC improves the coding efficiency by representing the strings of zeroes compactly. After the zig-zag scan ±1 are the highest nonzero coefficients CAVLC signals the number of high-frequency ±1 coefficients ('Trailing Ones') in a compact way. The neighbouring blocks' number of nonzero coefficients is correlated; therefore, the number of coefficients is encoded based on a look-up table which varies for a different number of nonzero coefficients in the neighbouring blocks. The magnitude of nonzero coefficients (level) tends to be larger at the start of the reordered array (near the DC coefficient) and smaller towards the higher frequencies. CAVLC takes advantage of this by adapting the choice of VLC look-up table for the level parameter depending on recently-coded level magnitudes. CAVLC encoding of a block of transform coefficients proceeds as follows:

1. Encode the number of coefficients and trailing ones (coeff-token)

1.1 Compute the coeff-token which is equal to the total number of nonzero coefficient (TatolCoeffs) and the number of trailing ±1 values

- TotalCoeffs can be anything from 0 (no coefficients in the 4×4 block) to 16 (16 nonzero coefficients)
- TrailingOnes can be anything from 0 to 3. If there are more than three trailing ±1s, only the last three are treated as 'special cases' and any others are coded as normal coefficients

1.2 Select the look-up table based on variable nC as a function of the number of nonzero coefficients in the left-hand and upper previously coded blocks (nA and nB respectively) and using Table 3.5 in order to determine the look-up table.

- If upper and left hand previously coded blocks are both available, B and A are both available: nC = round ((nA + nB)/2
- Else if only the upper is available, nC = nB;
- Else if only the left block is available, nC= nA;
- Else if neither is available, nC = 0

Table 3.5: Choice of look-up table for coeff_token

nC	Table for coeff-token
0,1	Table 1
2,3	Table 2
4,5,6,7	Table 3
8 or more	Table 4

1. Encode the sign of each TrailingOne

   2.1 For each TrailingOne (trailing ±1) signaled by coeff-token, the sign is encoded with a single bit (0 = +, 1 = −) in reverse order, starting with the highest-frequency TrailingOne.

2. Encode the levels of the remaining nonzero coefficients

   The level (sign and magnitude) of each remaining nonzero coefficient in the block is encoded in reverse order, starting with the highest frequency and working back towards the DC coefficient.

2.1 The code for each level is made up of a prefix (level prefix) and a suffix (level suffix). The length of the suffix (suffixLength) may be between 0 and 6 bits and suffixLength is found as follows:

- Initialise suffixLength to 0 (unless there are more than 10 nonzero coefficients and less than three trailing ones, in which case initialise to 1).
- Encode the highest-frequency nonzero coefficient.
- If the magnitude of this coefficient is larger than a predefined threshold, increment suffixLength. (If this is the first level to be encoded and suffixLength was initialised to 0, set suffixLength to 2).

The thresholds are listed in Table 3.6; the first threshold is zero which means that suffixLength is always incremented after the first coefficient level has been encoded.

Table 3.6: Thresholds for determining whether to increment suffixLength

Current sffixLenght	Threshold to increment suffixLenght
0	0
1	3
2	6
3	12
4	24
5	48
6	N/A

3. Encode the total number of zeros before the last coefficient. The sum of all zeros preceding the highest nonzero coefficient in the reordered array is coded with a VLC, total zeros. This approach means that zero-runs at the start of the array need not be encoded.
4. Encode each run of zeros

5.1 The number of zeros preceding each nonzero coefficient (run before) is encoded in reverse order. A run before the parameter is encoded for each nonzero coefficient, starting with the highest frequency, with two exceptions:

- If there are no more zeros left to encode (i.e.[run before]=total zeros), it is not necessary to encode any more run before values.
- It is not necessary to encode run before for the final (lowest frequency) nonzero coefficient.

5.2 The VLC for each run of zeros is chosen depending on:

- (a) the number of zeros that have not yet been encoded (ZerosLeft)

- (b) the run before. For example, if there are only two zeros left to encode, run before can only take three values (0, 1 or 2) and so the VLC need not be more than two bits long. If there are six zeros still to encode then run before can take seven values (0 to 6) and the VLC table needs to be correspondingly larger.

Chapter 4

A New Adaptive Motion Estimation Process

In Chapter 1 the function of a typical motion estimation system was demonstrated and it was seen that for each inter-mode block in the current frame the best matching block is selected within a search window in its reference frame(s) based on the energy of the block (e.g. The Mean Square Error - MSE, Mean Absolute Difference - MAD, Sum of Absolute Differences - SAD, or cross-correlation). Therefore, at the end of the motion estimation process for each inter-mode block there exists only one matching block that results in a residual block of minimum energy and a single two-dimensional motion vector. In our proposed method, in order to benefit a wider range of options for motion vector curve fitting as described, instead of generating only one best matching block with one motion vector, a set of matching blocks are chosen to form a set of matching block candidates for each block. The elements of each matching block set are the blocks within the search window in the reference frame(s) that the energy of each of the blocks is less than a predefined threshold. In other words, in the motion estimation process for an inter-mode block a set of matching block candidates are found within the search window that each candidate has an energy value (e.g. MSE, MAD, SAD, cross-correlation) less than a predefined

mode energy threshold and also has an estimated bitrate less than a determined bitrate threshold. The value of the bitrate threshold affects the operation of the proposed method in terms of the number of bits needed to encode the residual data block. The number of bits for encoding the residual block directly or indirectly depends on the energy of the block. Therefore, if the threshold is not chosen properly it may lead to the selection of a matching block candidate that results in large number of bits required to encode the residual block even if the motion vector of this block is a potential motion vector to be used for the motion vector curve fitting algorithm. In the next sections the following will be discussed: (1) the rate control problem that deals with estimating the number of bits required for encoding a quantized-transformed block as a function of the quantization parameter and the block characteristics such as its energy or the number of non-zero quantized-transformed coefficients; (2) the optimum threshold value is investigated based on a rate control model in order to bound the bitrate of the selected matching block candidates to approximately the bitrate needed to encode the residual block candidate with the minimum energy.

4.1 Rate Control Concept and its Application in Video Coding Standards

A rate control algorithm dynamically adjusts encoder parameters to achieve a target bitrate. It allocates a budget of bits to each group of pictures, individual picture, and/or sub-picture in a video sequence. Rate control is not a part of the H.264 standard, but the standards group has issued non-normative guidance to aid in implementation. Block-based hybrid video encoding, schemes such as the MPEG [38] and H.26* [19] families, achieve compression not only by removing truly redundant information from the bitstream, but also by making small quality compromises in ways that are intended to be minimally perceptible. In particular, the quantization parameter (QP) regulates how much spatial detail is retained. When the QP is

very small, almost all that detail is retained. As QP is increases, some of that detail is aggregated so that the bit rate drops – but at the cost of an increase in distortion and a decrease in quality , Figure 4.1a. As source complexity varies during a sequence, we move from one such curve to another, Figure 4.1b.

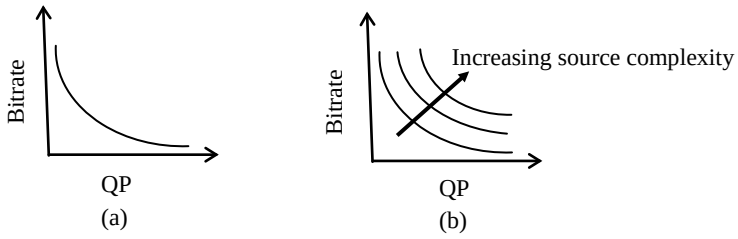
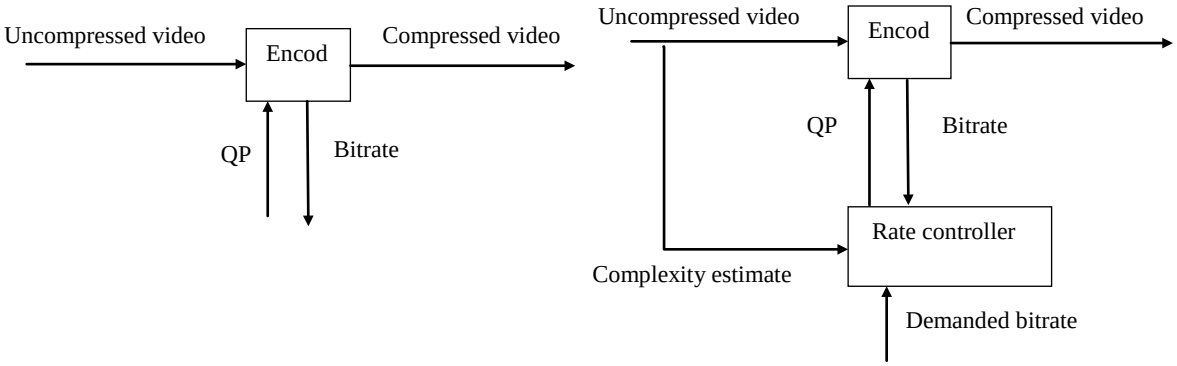


Figure 4.1: Increasing distortion and decreasing quality.

The diagram in Figure 4.2a illustrates the open loop (or variable bit rate) operation of a video encoder. The user supplies two key inputs: the uncompressed video source and a value for QP.

As the source sequence progresses, compressed video of a fairly constant quality is obtained, but the bitrate may vary dramatically; this is because the complexity of the pictures is continually changing in a real video sequence, it is not so obvious what value of QP to select. If we fix QP for an "easy" part (i.e. slow motion and uniform areas) of the frame sequence then the bit rate will go up dramatically when the "hard" (i.e. frame sequences with more motion) parts are reached. In reality, constraints imposed by the decoder buffer size and network bandwidth force us to encode video at a nearly constant bitrate. The diagram in Figure 4.2b suggests that we must dynamically vary QP based upon estimates of the source complexity, so that each picture (or group of pictures) receives an appropriate allocation of bits to work with. Rather than specifying QP as input, the user specifies their required bitrate instead. Therefore, rate control is a technique to achieve the target bitrate by regulating the quantization parameter.



(a) Open loop encoding (b) Close loop encoding

Figure 4.2: Open and Close loop encoding.

In our case we want to make use of the rate control concept to predict the number of bits needed for encoding a residual block for a specific QP, therefore another aspect of the rate control model is used in this work. Since we are encoding the whole picture with a fixed quantization parameter, therefore based on the rate control model we can predict the number of bits needed to encode the block using the block's complexity and the value of the quantization parameter. Therefore, we define a way to set the threshold for selecting the blocks with the motion vectors that are the motion vector curve fitting process candidates. We choose blocks where their estimated number of bits for encoding their residual blocks is close enough to the actual number of bits needed for encoding the block with minimum MSE that is selected as the H.264 motion estimation selection. The method of setting the threshold based on rate control model is described in detail in Section 4.2.

The methods of predicting the number of bits to encode a data block is mainly based on the statistical model chosen for the block's elements. Three main procedures, linear, mathematical, and the rate distortion model (based on the rate distortion theory) are used to predict the number

of bits needed to encode a quantized-transform block data before transformation and quantization. Although these methods are different in terms of their linearity of the distortion procedures, they all depend upon the Probability Distribution Function (PDF) that is selected for the block data. It should be noted that the concept of the number of bits needed to encode only a block can be expanded to the number of bits needed to encode a larger unit such as a slice or frame; however the only difference is the statistical characteristic of the selected PDF, such as the data variance and mean, relate to the whole unit.

4.1.1 Rate Distortion Models Based on Rate Distortion Theory

Based on the rate distortion theory [39] for a given source model distribution and specific distortion definition there exists a rate model that is a function of total distortion and the source statistical characteristics such as its variance or mean:

$$R(D) = \max[sD + \sum_{u} Q(u)\ln(\lambda(u))]$$
$$\Lambda_S \equiv \left\{\lambda(u) : \sum_{u} \lambda(u)Q(u)e^{sd(u,v)} \le 1\right\} \tag{4.1}$$

Where R(D) is the estimated number of bits needed to encode the source samples after distortion, Q(u) is a given source distribution model, D is the distortion function, Equation (4.2), λ(u) is a Lagrangian function, Equation (4.3), d(u,v) is a measure of distortion (e.g. |u-v| or $|u-v|^2$ or quantization parameter), and s is the model parameter.

$$D = \sum_{u}\sum_{v} \lambda(u)\,p(v)e^{sd(u,v)} \tag{4.2}$$

$$\lambda(u) = \left(\sum_{v} p(v)e^{sd(u,v)}\right)^{-1} \tag{4.3}$$

where p(v) is the source model distribution after distortion, for instance source model distribution after quantization, which can be the same as the original source model distribution with different statistical characteristics.

Based on Equation (4.1), so far various rate control models have been proposed for different source distribution models; for example some of the proposed rate control functions are based on modeling the source samples as Gaussian distributed variables [40, 41, and 42] or as Laplacian distributed samples [43 and 44]. The exponential distribution is exploited in work by Tian et al [45] and Li et al [46] to present the relationship between bitrate and distortion. Table 4.1 shows the rate control modeled formulas that have been proposed based on the three main source model distributions. The distortion, D, in the formulas in Table 4.1 is replaced by a factor of squared quantization step size [41, 43], or by a combination of quantization step sizes with distortion measures such as MAD, [44, 45, 47]. In some applications the direct forms of the rate models in Table 4.1 are used to predict the rate; however, in some applications they are transformed into more applicable and practical forms [45, 42]. In the H.264/AVC [19] video coding, the quadratic R-D model has been employed based on the assumption that the residual signal after DCT transformation obeys a Laplacian distribution.

Table 4.1: The rate models based on different probability distributed source models[10].

Probability density function	Q(u)	Bit rate formulation
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{u^2}{2\sigma^2}}$	$R(D) = \frac{1}{2}\ln\frac{\sigma^2}{D}$
Laplacian	$\frac{1}{\sqrt{2}\sigma} e^{-\frac{\sqrt{2}\lvert\mu-u\rvert}{\sigma}}$	$R(D) = \frac{1}{2}\log\frac{\sigma^2}{D}$
Exponential	$\frac{1}{2\mu} e^{-\frac{\lvert u\rvert}{\mu}}$	$R(D) = -\ln(\frac{D}{\mu})$

It is derived according to Taylor's series expansion which removes the high-order infinitesimally small values. The current rate control proposal [54] for H.264/AVC is composed of a quadratic model and a linear tracking model. The quadratic model is described as: (as described in Section 4.1.2). The quadratic model is described as:

$$R(D) = \frac{a_1 s}{Q_{step}} + \frac{a_2 s}{Q_{step}^2} \tag{4.4}$$

where R denotes the number of bits, a_1 and a_2 are model parameters, Q_{step} indicates the quantization step, and S is the MAD of the current block.

4.1.2 Linear Rate Distortion Model

Assuming that each of the transform coefficients ($x_{l,k}$) in a m×n transformed residual block is Laplacian distributed; therefore, each element (k, 1) of the DCT block will have the following PDF:

$$f(x_{k,l}) = \frac{1}{2\alpha_{k,l}} e^{-\frac{\lvert x_{k,l}\rvert}{\alpha_{k,l}}} \qquad for\ \ \mathrm{m} < k, l < n \tag{4.5}$$

[10] μ and σ are mean and variance respectively.

The entropy of the entire block is given as the sum of the entropies of each coefficient separately:

$$H(x) = \sum_{k,l} H_{k,l} = \sum_{k,l} \ln 2e\alpha_{k,l} \tag{4.6}$$

where the parameter $\alpha_{k,l}$ is estimated by calculating the mean absolute value of a number of samples (e.g. taken from a series of DCT blocks in a slice) of each coefficient. Based on information theory, entropy is usually expressed as the average number of bits needed to store or communicate one symbol, based on this fact the entropy of the block can be used as an estimator for the number of bits required to encode quantized-transformed residual data block. Instead of considering only one block we can expand the block entropy to the slice entropy, however, the slice length entails a trade-off between PDF uniformity for the slice and the $\alpha_{k,l}$ reliability. The number of bits for encoding a block of data after quantization can be estimated as:

$$R(s) = m(s)H(x) + b(s) \tag{4.7}$$

where m(s) is a measure of distortion caused by quantization which can be the quantization step (s) of a block and b(s) is the number of bits needed for encoding the block header information. In some rate models the quantization step is chosen as the distortion measure because of its computational simplicity. The MSE or MAD can be used and the same formulation is still valid since they are also a function of quantization.

The principle Equation (4.3) means that for every block a linear model predicts the rate as a function of the parameter H(x) for every possible quantization step size s. The parameters of the linear models [m(s) and b(s)] are experimentally determined for a representative set of video sequences, coded at all possible step sizes. The rate model in H.261 and H.263 [5] is based on Equation (4.3) which is equal to:

$$R(Q) = \frac{X}{QP} + L \tag{4.8}$$

Where X and L are the model parameters, where L is a constant. Here we can see that the distortion measure is chosen as the quantization step size and X represents the entropy of the residual data block.

4.1.3 Mathematically-Based Rate Distortion Models

In these schemes generally a mathematical model is proposed and the model parameters are found from implementing an extensive series of tests, then the model with its parameters is generalized to video sequences with the same characteristics. For instance a curve-fitting method based on experimental data can be applied [48] to design the rate model and the distortion models separately as a function of the energy of the residual block which is the block variance. In other methods [49], two simple linear functions for rate (R) and distortion (D) as function of e quantisation step size are proposed and the functions' parameters are estimated by using Kalman filtering.

4.1.4 ρ-Domain Rate Modeling Scheme

These methods are based on modeling the rate and distortion as a function of the number zeros among a DCT block after quantization [50-55]. The basis of this idea is that after the DCT coefficients are quantized with a quantization parameter q, let ρ be the percentage of zeros among the quantized coefficients, ρ monotonically increases with q. Therefore, assuming that the distribution of the transform coefficients is continuous and positive [50], there is a one-to-one mapping between ρ and q. This implies that, mathematically, rate (R) and distortion (D) are also functions of ρ, denoted by R (ρ) and D (ρ). A study of the rate and distortion as functions of ρ is called ρ-domain analysis.

4.2 Determining the Threshold for Selecting Candidate Blocks

For selecting the candidate matching blocks based on the number of bits needed for encoding their corresponding residual blocks we use the rate control model based on the model presented in Equation(4.4), but for simplicity it is modified as Equation (4.9).

$$R = \frac{a_1 MAD}{Q_{step}}$$

$$MAD = \frac{\sum_{i,j} \left| Residual_{i,j} \right|}{M \times N} \qquad \text{for } 0 < i < M \text{ and } 0 < j < N \tag{4.9}$$

Where M and N are the height and width of the residual block respectively; i and j are the block references. For selecting the candidate blocks we propose a 6-step motion estimation algorithm as follows:

1. Find the best matching block within the search window (the best matching block is considered as the one that has the least MSE).
2. Encode its quantized-transformed residual block and compute the number of bits used to encode the block ($R_{threshold}$).
3. Solve Equation (4.9) using $R_{threshold}$, Q_{step} and MAD of the best matching block found in Step 1 ($MAD_{threshold}$), to find the model parameter a_1.

$$a_1 = \frac{R_{threshold} \times Q_{atep}}{MAD_{threshold}} \tag{4.10}$$

4. Compute the MAD of each candidate block within the search window ($MAD_{current}$), if it is equal or less than Threshold×$MAD_{threshold}$ (Threshold is determine in the next step) it shows that the estimated bitrate $R_{estimate}$, Equation (4.9), for this block is less than Threshold×$R_{threshold}$ (the variables MAD and R are in direct relation to the values in

Equation(4.9)) this block is selected as a candidate since it is based on the rate model it shows that the selected block needs a lower number of bits to be encoded than the number of bits needed to encode the selected block with the minimum MSE multiplied by the amount of threshold.

$$R_{estimate} = \frac{a_1 MAD_{current}}{Q_{step}} \tag{4.11}$$

$$\text{If } \mathrm{MAD}_{\text{current}} < (\text{Threshold}) \times \mathrm{MAD}_{\text{threshold}} \Rightarrow \mathrm{R}_{\text{estimate}} < (\text{Threshold}) \times \mathrm{R}_{threshold}$$

In order to find the Threshold the following steps are required:

1. Four test video sequences Foreman, Carphone, Miss America and Suzie are encoded via the same proposed curve fitting algorithm, except in the adaptive motion estimation system the matching block candidates are selected based only on their MSE values (for each block in the current frame up to 16 matching blocks are selected while the selected matching blocks have the least MSE values and below a predefined threshold), see Tables (4.2-4.5).

2. The difference between the number of bits used to encode the curve fitting algorithm motion data (MVBit_Curve) and the number of bits required for encoding the motion vector differences (encoded motion vectors via the H.264 algorithm), MVBit_H.264, are computed via actually entropy encoded data.

$$\Delta MVBit = MVBit\_H.264 - MVBit\_Curve \tag{4.12}$$

3. The difference between the number of bits for encoding residual data which results from the proposed curve fitting algorithm (DCTBit_Curve) and the H.264 standard (DCTBit_H.264) are computed using Equation 4.13.

$$\Delta DCTBit = DCTBit\_Curve - DCTBit\_H.264 \quad (4.13)$$

(Note that the residual blocks of the minimum MSE value are expected to have a lower number of bits than the blocks selected via the curve fitting algorithm; therefore, it is expected that the ΔDCTBit is a positive value).

4. In order to improve the total bitrate via the curve fitting algorithm it is obvious that the amount of decrease in the motion vector bitrate (ΔMVBit) should be larger than the increase in the DCT residual bitrate (ΔDCTBit);

$$\Delta(\text{DCTBit})\% = \frac{\Delta(\text{DCTBit})}{(\text{DCTBit\_H}.264)} \times 100 \quad (4.14)$$

$$\alpha = \frac{\Delta(\text{DCTBit})}{\Delta(\text{MVBit})} \quad (4.15)$$

$$\text{if} \quad \alpha < 1 \qquad \text{Threshold} = \Delta(\text{DCTBit})\% \quad (4.16)$$

$$\text{else} \qquad \text{Threshold} = \frac{\Delta(\text{DCTBit})\%}{1+\alpha} \quad (4.17)$$

Therefore, in situations where the increase in the number of residual bitrate counteracts the decrease in the number motion vector bits, the maximum allowed residual bitrate (Threshold) is set to the ratio defined in Equation (4.17). Examining Table 4.2 to Table 4.5, it is observed that, for encoded test video sequences that correspond to $\alpha > 1.0$, (the underlined bolded α values in the tables) the decrease in motion vector bits, (ΔMVBit), is smaller than the increase in the residual bitrate, (ΔDCTBit).

Therefore, the percentage of the increase in the number of residual bits (ΔDCTBits%) should be decreased and it is bounded by the threshold value equal to Threshold as defined in Equation (4.17) .

Table 4.2: Residue bits comparison using Foreman test video

	Block size(4×4) (ΔMVBit) = 223715				Block size(8×8) (ΔMVBit) = 33334				Block size(16×16) (ΔMVBit) = 3291			
QP	H.264	AME	(ΔDC TBits)	A	H.264	AME	(ΔDC TBits)	α	H.264	AME	(ΔDC TBits)	α
6	5455039	5647927	3.536	0.9	3396256	3423393	0.799	0.8	1736212	1737840	0.094	0.5
12	3342859	3545316	6.056	0.9	2095411	2124996	1.412	0.9	1069958	1072704	0.257	0.9
18	1450721	1599572	10.26	0.7	950393	970549	2.121	0.6	486946	489300	0.483	0.8
24	983133	995537	1.262	0.1	613108	617085	0.649	0.1	322200	322951	0.233	0.2

Table 4.3: Residual bits comparison using Carphone test video

	Block size(4×4) (ΔMVBit) = 1071862				Block size(8×8) (ΔMVBit) = 162980				Block size(16×16) Δ(MVBit) = 12971			
QP	H.264	AME	Δ(DC TBits)	A	H.264	AME	Δ(DC TBits)	α	H.264	AME	Δ(DC TBits)	α
6	6316374	6564304	3.925	0.2	5475438	5561072	1.56	0.5	3993394	4024014	**0.767**	**2.4**
12	3432054	3700127	7.812	0.2	3132453	3225412	2.968	0.6	2323899	2357615	**1.451**	**2.6**
18	1555587	1707540	9.768	0.2	1368746	1437196	5.001	0.4	1023770	1050563	**2.617**	**2.0**
24	1313102	1329019	1.212	0.0	1053221	1066190	1.232	0.1	760322	766013	0.748	0.4

Table 4.4: Residual bits comparison using Miss America test video

	Block size(4×4) Δ(MVBit) = 688639				Block size(8×8) Δ(MVBit) = 180347				Block size(16×16) Δ(MVBit) = 20041			
QP	H.264	AME	Δ(DC TBits)	A	H.264	AME	Δ(DC TBits)	α	H.264	AME	Δ(DC TBits)	α
6	5409512	5629296	4.239	0.3	6217240	6285657	1.100	0.4	6200332	6217248	0.273	0.8
12	2706479	2927952	8.183	0.3	3456479	3538578	2.375	0.5	3598931	3615149	0.45	0.8
18	1476772	1537731	4.128	0.0	1548893	1600314	3.320	0.3	1528618	1555695	1.77	**1.3**
24	1277041	1283841	0.532	0.0	1226833	1234861	0.65	0.0	1163818	1169975	0.561	0.3

Table 4.5: Residual bits comparison using Suzie test video

	Block size(4×4) Δ(MVBit) = 640177				Block size(8×8) Δ(MVBit) = 28446				Block size(16×16) Δ(MVBit) = 2359			
QP	H.264	AME	Δ(DC TBits)	α	H.264	AME	Δ(DC TBits)	α	H.264	AME	Δ(DC TBits)	α
6	4374307	4554773	4.126	0.3	3788567	3854270	0.799	**2.3**	3133122	3180306	**1.5**	**20.0**
12	2328214	2513811	7.972	0.3	1999850	2076152	1.412	**2.7**	1626424	1681157	**3.37**	**23.2**
18	1149917	1215167	5.674	0.1	977783	1009292	2.121	**1.1**	821638	843767	**2.69**	**9.3**
24	941351	951020	1.027	0.0	797427	803107	0.649	0.2	671317	675622	**0.64**	**1.8**

In order to make the curve fitting algorithm independent of the video sequence type the thresholds for each block size and quantization parameter are computed as the mathematical mean of the threshold values of all tested video sequences. Evaluating the threshold values in Table 4.6 to Table 4.9 the following are observed:

1. It is observed that larger block size motion estimation results in smaller threshold values since as the block size increases the number of motion vectors decreases leading to a smaller ΔMVBit or a larger α value.

2. The threshold values increases as the quantization parameter increases for the first three quantization parameters for the tested video sequences and three block sizes. This is justified by the fact that as the quantization parameter increases the number of residual bits decreases which leads to a smaller ΔDCTBit.

Table 4.6: Residual bit increase threshold values for Forman test video

QP	Block size(4×4) Threshold value%	Block size(8×8) Threshold value%	Block size(16×16) Threshold value%
6	3.5360	0.7990	0.0938
12	6.0564	1.4119	0.2566
18	10.2604	2.1208	0.4834
24	1.2617	0.6487	0.2331

Table 4.7: Residual bit increase threshold values for Carphone test video

QP	Block size(4×4) Threshold value%	Block size(8×8) Threshold value%	Block size(16×16) Threshold value%
6	3.925	1.5640	***0.2282***
12	7.8109	2.9676	***0.4031***
18	9.7682	5.0009	***0.8537***
24	1.2122	1.2314	0.7485

Table 4.8: Residual bit increase threshold values for Miss America test video

QP	Block size(4×4) Threshold value%	Block size(8×8) Threshold value%	Block size(16×16) Threshold value%
6	4.2392	1.1004	0.2728
12	8.1831	2.3752	0.4506
18	4.1279	3.3198	***0.7534***
24	0.5325	0.6544	0.5606

Table 4.9: Residual bit increase threshold values for Suzie test video

QP	Block size(4×4) Threshold value%	Block size(8×8) Threshold value%	Block size(16×16) Threshold value%
6	4.1256	***0.5240***	***0.0717***
12	7.9717	***1.0361***	***0.1390***
18	5.6743	***1.5289***	***0.2594***
24	1.0271	0.71229	***0.2270***

Chapter 5

Simulation and Results

In Chapters 3 and 4 the proposed method was described and we saw that the motion vectors of collocated blocks along a video sequence are fitted into a curve such that the curve's control points (Category One), "non-reconstructable" motion vectors from the curve (Category Four), and Category Three and Four indices are enough to reconstruct the motion vectors at the decoder side. In the present chapter the potential of collocated blocks for being encoded via our curve fitting scheme is examined and the results of the simulations on several video sequences are presented in detail. The efficiency of the proposed method in comparison with a previously proposed method in the widely used H.264 scheme is explained in terms of the bitrate saving in encoding motion vectors is evaluated. To evaluate the coding performance of the proposed method the test conditions listed in Table 5.1 are applied in the simulations using four test video sequences: Foreman, Carphone, Miss America and Suzie. These sequences are chosen as they contain different types of motion; Foreman and Carphone sequences represent relatively complex motions, the Foreman sequence specifically has mainly little background motion and a lot of foreground motion. Suzie and Miss America sequences contain moderate and slow motions

respectively. Three block sizes, (4×4), (8×8) and (16×16) were used for the motion estimation process. The smallest block size, (4×4), has been chosen in order to evaluate the performance of proposed method for small block size motion estimation which generates a significant amount of motion vectors while providing higher video quality. We make use of the ¼ pel motion accuracy motion estimation conforming to the H.264 Baseline profile.

Table 5.1: Experimental conditions.

Sequence Size	QCIF (176 ×144)			
Sequence Name	Foreman	Carphone	Miss America	Suzie
Frame Count	200	200	150	150
Frame Rate	30Hz			
MV Search	Full Search ± 32 pixels			
QP	6, 12, 18, 24			
Coding Options	1 reference pictures Fixed block sizes ME; (4×4), (8×8), (16×16) 1/4 –pel ME accuracy 1 picture = 1 slice			

5.1 Adaptive Motion Estimation Process Evaluation

In this section the functionality of the proposed Adaptive Motion Estimation (AME) process (Chapter 4) is explained by considering the variations in the bitrate of the residual blocks and the picture quality compared with the reference (H.264 standard). The bitrate in this case is defined as the number of bits used to encode the residual blocks per second which itself depends on the

applied video frame rate (that, in this experiment, is equal to 30fps). A Peak Signal to Noise Ratio (PSNR) is chosen as the quality measurement metric, Equation (5.2). PSNR is most easily defined via the mean squared error (MSE) which for two M×N images I and K where one of the images is considered a noisy approximation of the other is defined as:

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)] \tag{5.1}$$

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{5.2}$$

5.1.1 Residual Data Bitrate Comparison

In order to discuss the efficiency of the proposed motion vector encoding scheme compared with the H.264 standard, it is important to compare the rate of the change in the motion vector bitrate while the number of bits needed to encode the residual blocks from both methods are the same. Since, in the proposed method the selected matching blocks in the AME method are not necessarily the same as the blocks of the minimum MSE (or best matching blocks); therefore, the number of bits needed to encode the residual blocks resulting from two methods are not equal (although, as mentioned in Chapter 4, the amount of variation in the number of bits for encoding residual blocks in the AME is limited by applying a rate control technique). Hence, the amount of increase or decrease in the residual block bits is considered in computing the motion vector bitrate. For example, if the matching block selected from the AME method leads to an increase of 20 bits, this amount is added to the number of bits required for encoding motion data by the proposed encoding method so the motion vector bitrates are compared by the same number of bits needed to encode residual blocks. The residual blocks have been chosen constrained by two criteria: (1) the initial energy threshold which determines the block mode, inter or intra, and (2)

the bitrate threshold which is compared with the estimated bitrate for the residual block. Table 5.2 to Table 5.5 show the variation of the number of residual bits for the four tested video sequences; the Δ(Bits)% is the amount of increase in the number of bits for encoding residual blocks via our AME method in comparison with the number of bits needed to encode the residual via the H.264 algorithm such that this increase in the number of bits would be compensated by the motion vector bitrate improvement. The thresholds determined based on the tests results are explained in Chapter 4. As the results show, the variations of the actual bitrates are bound to less than the determined bitrate thresholds with precision to the first decimal place. It will be shown in Section 5.3 that a precision to the first decimal place was obtained in the bitrate estimation of residual blocks by the rate control model is sufficient, as it leads to a total bitrate decrease in encoding motion vectors via our proposed curve fitting scheme.

Table 5.2: Comparison of Residual Bits Using Foreman Test Video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%
6	5455039	5647927	3.536	3396256	3423393	0.799	1736212	1737840	0.0938
12	3342859	3545316	6.056	2095411	2124996	1.412	1069958	1072704	0.257
18	1450721	1599572	10.260	950393	970549	2.121	486946	489300	0.483
24	983133	995537	1.262	613108	617085	0.649	322200	322951	0.233

Table 5.3: Comparison of Residual Bits Using Carphone Test Video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%
6	6316374	6564304	3.925	5475438	5561072	1.564	3993394	4001917	0.213
12	3432054	3700127	7.812	3132453	3225412	2.968	2323899	2333194	0.400
18	1555587	1707540	9.768	1368746	1437196	5.001	1023770	1032587	0.861
24	1313102	1329019	1.212	1053221	1066190	1.23	760322	766013	0.7485

Table 5.4: Comparison of Residual Bits Using Miss America Test Video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%
6	5409512	5629296	4.239	6217240	6285657	1.100	6200332	6217248	0.273
12	2706479	2927952	8.183	3456479	3538578	2.375	3598931	3615149	0.451
18	1476772	1537731	4.128	1548893	1600314	3.320	1528618	1538620	0.654
24	1277041	1283841	0.532	1226833	1234861	0.654	1163818	1169975	0.560

Table 5.5: Comparison of Residual Bits Using Suzie Test Video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%	H.264	AME	Δ(Bits)%
6	4374307	4554773	4.126	3788567	3808441	0.525	3133122	3135171	0.065
12	2328214	2513811	7.972	1999850	2020600	1.038	1626424	1628697	0.140
18	1149917	1215167	5.674	977783	989603	1.209	821638	823818	0.265
24	941351	951020	1.027	797427	803107	0.647	671317	672817	0.223

The average bitrate estimation errors versus the quantization parameters are presented in Figure 5.1 for the four video sequences. The average error is computed as the mathematical mean of the errors resulting from estimating the bitrates for three tested block sizes for each quantization parameter:

$$Error = \sum_{j=1}^{4} \sum_{i=1}^{3} (error_i) \tag{5.1}$$

where $error_i$ is equal to:

$$\text{if } Threshold - \Delta(Bit)\% > 0 \qquad error_i = Threshold - \Delta(Bit)\% \tag{5.2}$$
$$\text{else} \qquad error_i = 0$$

where i and j are the block size and video sequence indexes respectively. Since the blocks are chosen such that their estimated bitrates should be less than the threshold, the error is only considered for the block sizes which result in the actual bitrate larger than the threshold. The maximum average error is less than 0.0045% of the total bitrate in all cases.

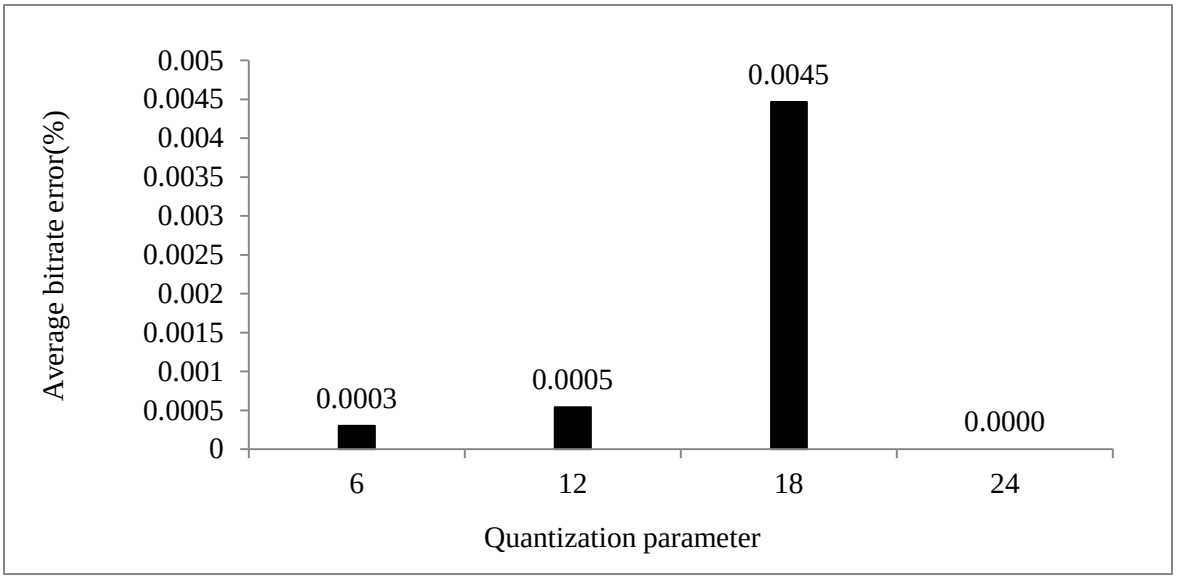


Figure 5.1: Percentage of the average bitrate estimation error versus quantization parameter.

5.1.2 Picture Quality Evaluation Using AME System

Motion estimation systems with dissimilar decision making criteria generate coded videos of different picture quality values; therefore, it is necessary to evaluate each system in terms of the quality it leads to rather than the system itself. Table 5.6 to Table 5.9 contain the average PSNR values of four encoded test video sequences from H.264 and our AME motion estimation method using block sizes (4×4), (8×8) and (16×16). The average PSNR is computed only for inter coded parts of the video sequences. It is important to consider that the intra coded areas in the videos coded via the standard and AME method are the same since, as explained in Chapter 3, it is based on the same predetermined MSE threshold in both algorithms; it is determined whether each block should be encoded in intra or inter modes. Therefore, for intra coded parts of the videos the PSNR value remains the same as those areas are encoded with the same method.

Examining the results from Tables (4.6-4.9), the question may be asked as to why the average PSNR of the larger block size motion estimations are larger than the average PSNR values resulting from smaller block sizes. The answer lies in the fact that in this case they are not comparable because the number of frames encoded in intra mode for different block size motion estimations are not equal since the average PSNR is computed by dividing the total PSNR values of the inter coded frames by the number of these frames in the video sequence. However, the reason that the PSNR is larger for larger block sizes is that as the block size increases the MSE between the current block and the blocks in the reference frames within the search window increases. Therefore, the MSE values are larger than the MSE threshold and the block is encoded in intra mode and the number of intra mode blocks may be increased to cover the entire frame. Hence, the number intra coded frames increases for larger block sizes and as a result the PSNR related to inter coded areas is divided by a smaller number leading to a larger PSNR.

Table 5.6: Picture quality comparison using Foreman test video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%
6	57.43	57.18	0.43	58.92	58.93	-0.00	61.80	61.82	-0.04
12	50.97	50.89	0.15	53.30	53.30	0.00	56.23	56.24	-0.03
18	47.72	47.47	0.52	49.94	49.84	0.19	52.63	52.64	-0.02
24	46.95	46.25	1.49	48.38	48.19	0.40	48.70	48.42	0.57

Table 5 . 7: Picture quality comparison using Carphone test video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%
6	57.71	57.69	0.03	58.95	58.95	0.00	61.816418	61.820	-0.00
12	51.39	51.29	0.20	53.47	53.46	0.02	56.33	56.34	-0.02
18	49.82	49.53	0.58	50.66	50.55	0.21	53.35	53.30	0.09
24	48.92	48.09	1.68	49.73	49.34	0.78	52.28	52.05	0.43

Table 5 . 8: Picture quality comparison using Miss America test video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%
6	57.54	57.32	0.38	58.93	58.93	0.00	56.98	56.98	0.00
12	51.88	51.69	0.36	53.54	53.53	0.00	51.59	51.61	-0.03
18	50.10	49.75	0.70	48.71	48.64	0.15	48.50	48.51	-0.01
24	49.53	48.80	1.47	50.53	50.19	0.66	52.47	52.30	0.32

Table 5 . 9: Picture Quality comparison using Suzie test video

	Block size(4×4)			Block size(8×8)			Block size(16×16)		
QP	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%	H.264	AME	Δ(PSNR)%
6	57.46	57.46	0.00	58.99	58.98	0.02	61.87	61.87	0.01
12	51.51	51.33	0.35	53.72	53.71	0.01	56.67	56.67	0.00
18	49.81	49.34	0.93	51.92	51.76	0.30	54.97	54.84	0.23
24	49.28	48.49	1.60	51.24	50.90	0.65	54.24	54.01	0.42

Analysing the results presented in the above tables showed the following conclusions:

- For larger block sizes the rate of decreasing picture quality resulting from encoding video sequences, using the AME method, decreases; the highest PSNR decrease is related to encoding the Carphone sequence ((4×4) block size AME (1.68% dB)), that is considered as a video sequence with more complex motions. Figure 5.2 depicts the average rate of

Δ(PSNR)% computed as the mean of Δ(PSNR)% for four quantization parameters versus the block size for the four tested video sequences.

- Using the (16×16) block size the AME method showed an improvement in PSNR of in the test results, up to 0.04514% dB for (16×16) block size AME using Foreman sequence

Therefore, better performance of the AME method is achieved for smaller quantization parameters with larger block sizes in comparison with the general motion estimation scheme. However, the PSNR variations are insignificant as the increase and decrease in PSNR resulting from the AME method are bounded to 0.823dB and 0.0279 dB respectively for the (4×4) block size AME using Carphone and for (16×16) block size AME using Foreman sequences respectively.

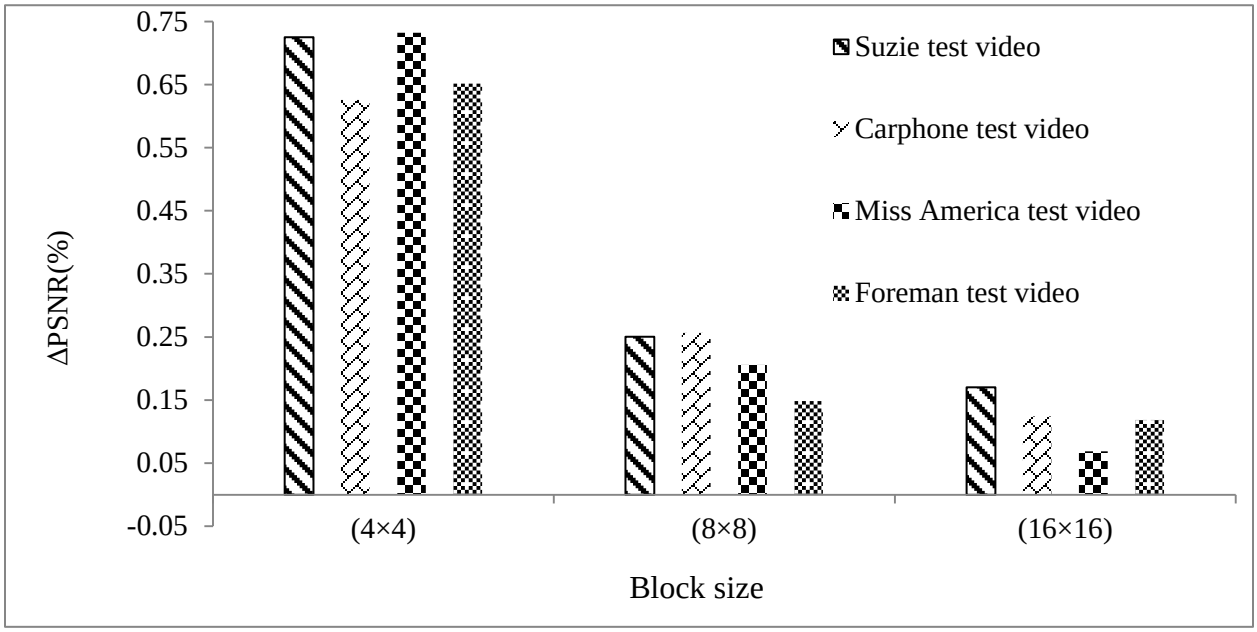


Figure 5.2: The average Δ(PSNR)% resulting from AME method for different block sizes.

5.2 Selecting the Two Optimum Categories Based on Category Statistics

In Section 3.3 it was explained that to reconstruct the motion vectors at the decoder side one of category indexes between One and Four and also between Two and Three should be selected and encoded and the other two categories due to the encoded video signal bitstream format are recognized automatically. Also, it mentioned that the least-occurring categories are chosen in order to save bitrate. The test results presented in Table 5.10 to Table 5.13 provide information about how the motion vectors (MV) of the four tested video sequences for three block sizes are distributed among the four categories after reconstruction from the curves. In other words it shows the percentage of the control points (key motion vectors belonging to Category One), the motion vectors that cannot be reconstructed from the curves (Category Four) and the motion vectors for which in order to generate their original values they should be rounded to their upper integer after reconstruction from the curve (Category Two), and vice versa (Category Three). This information is of great importance since it gives an overview of the functionality of the curve fitting algorithm for encoding motion vectors; examining the percentage of motion vector categories illustrates, by reduction of the number of motion vectors, that should be encoded after mapping motion vectors into a set of control points, Category Four motion vectors, and category indices.

Table 5.10: Distribution of category indices among the Motion Vectors of Foreman sequence

Block size	Four		Three		Two		One	
	MV number	Percentage	MV number	Percentage	MV number	Percentage	MV number	Percentage
(4×4)	155932	24.088	28240	4.362	90748	14.019	372416	57.531
(8×8)	16876	17.265	3952	4.043	51864	53.059	25056	25.633
(16×16)	1216	9.632	332	2.630	7796	61.755	3280	25.98

Table 5.11: Distribution of category indices among the Motion Vectors of Carphone sequence

Block size	Four		Three		Two		One	
	MV number	Percentage	MV number	Percentage	MV number	Percentage	MV number	Percentage
(4×4)	166560	19.210	32220	3.717	122916	14.177	545336	62.897
(8×8)	31672	18.565	8768	5.140	81024	47.494	49136	28.802
(16×16)	3388	10.951	832	2.690	19600	63.357	7116	23.002

Table 5.12: Distribution of category indices among the Motion Vectors of Miss America sequence

Block size	Four		Three		Two		One	
	MV number	Percentage	MV number	Percentage	MV number	Percentage	MV number	Percentage
(4×4)	195824	23.0971	28096	3.314	113176	13.349	510732	60.240
(8×8)	67084	33.319	10308	5.10	72436	35.978	51508	25.583
(16×16)	5052	10.661	1668	3.520	23020	48.578	17648	37.242

Table 5.13: Distribution of category indices among the Motion Vectors of Suzie sequence

Block size	Four		Three		Two		One	
	MV number	Percentage	MV number	Percentage	MV number	Percentage	MV number	Percentage
(4×4)	132116	21.23	24548	3.944	125728	20.203	339948	54.62
(8×8)	14632	11.293	5036	3.887	76144	58.768	33756	26.052
(16×16)	2428	9.007	748	2.775	19664	72.959	4112	15.257

Based on the results presented in Tables (5.10-5.13), the number of motion vectors belonging to Category Three, is less than the number of Category Two motion vectors for all test video sequences and all tested block sizes. On the other hand, the number of motion vectors belonging to the Category One (control points) is more than the Category Four motion vectors for all the test conditions except block sizes (8×8) in the video sequence Miss America. Figure 5.3 to Figure 5.6 illustrate the trends of the distribution of motion vectors for each video sequence and the tested block sizes. Therefore, one option would be to select Categories Four and Three as the selected category indices for (8×8) and (16×16) block size motion estimation methods and Categories Four and Two as the selected category indices for (4×4) block size motion estimation methods. However, we can choose the category indices based on the average percentage of each category proportion over the entire test conditions (as can be seen in Figure 5.7). On average 27.68% of motion vectors correspond to Category One and 13.02% of motion vectors correspond to Category Four (as shown in Figure 5.8). Therefore, the category indices for motion vectors of Category Four are encoded. On the other hand, on average, 31.48% and 2.82% of the motion

vectors belong to Category Two and Three respectively, as shown in Figure 5.1, which means encoding index category for the Category 3 motion vectors leads to more efficient compression.

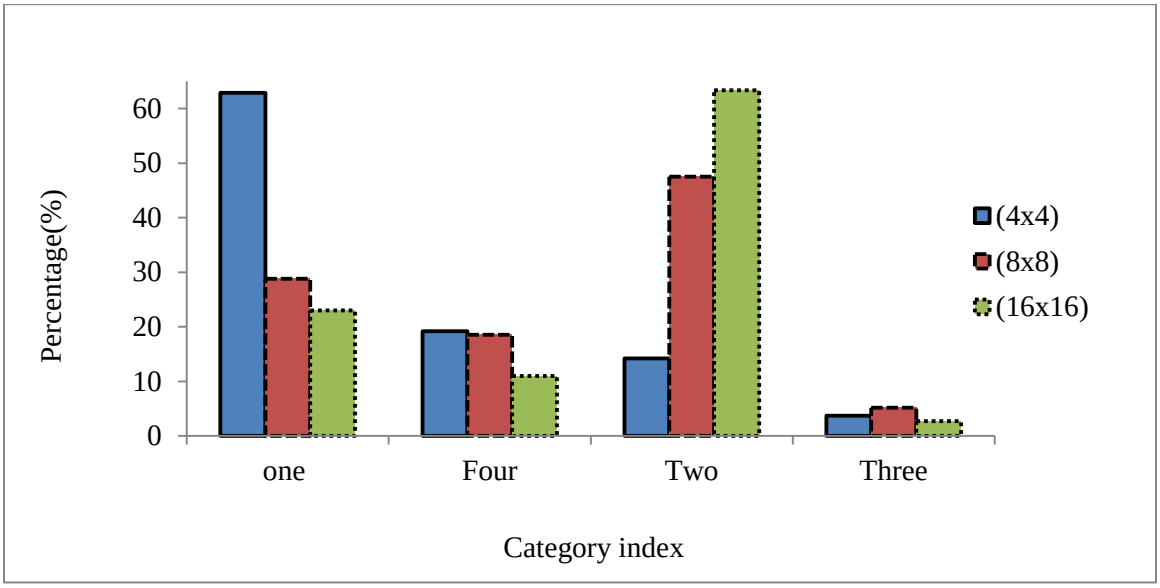


Figure 5.3: Distribution of Motion Vector Categories for Foreman Sequence

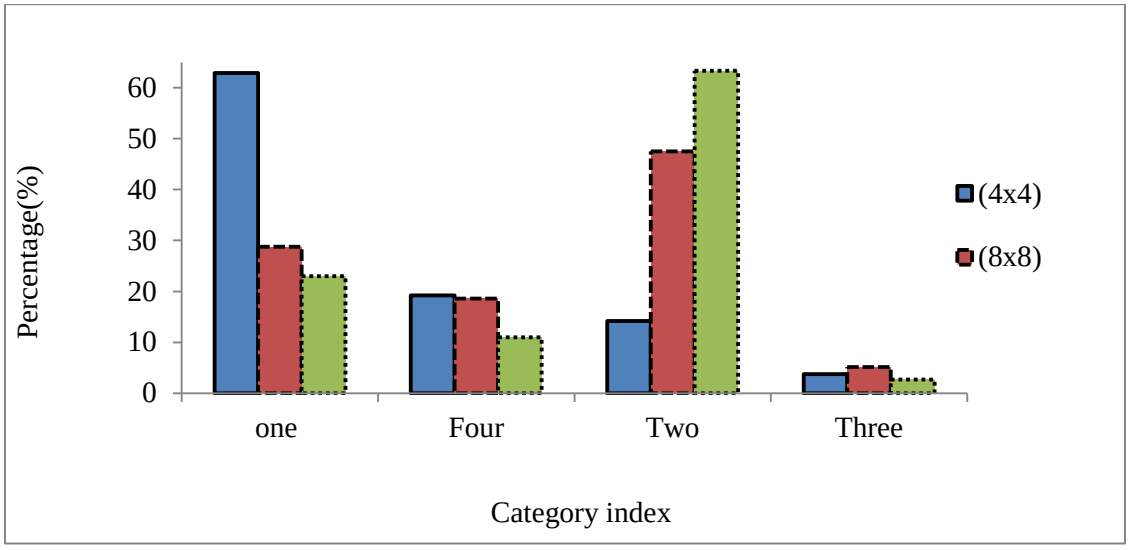


Figure 5.4: Distribution of Motion Vector Categories for Carphone Sequence

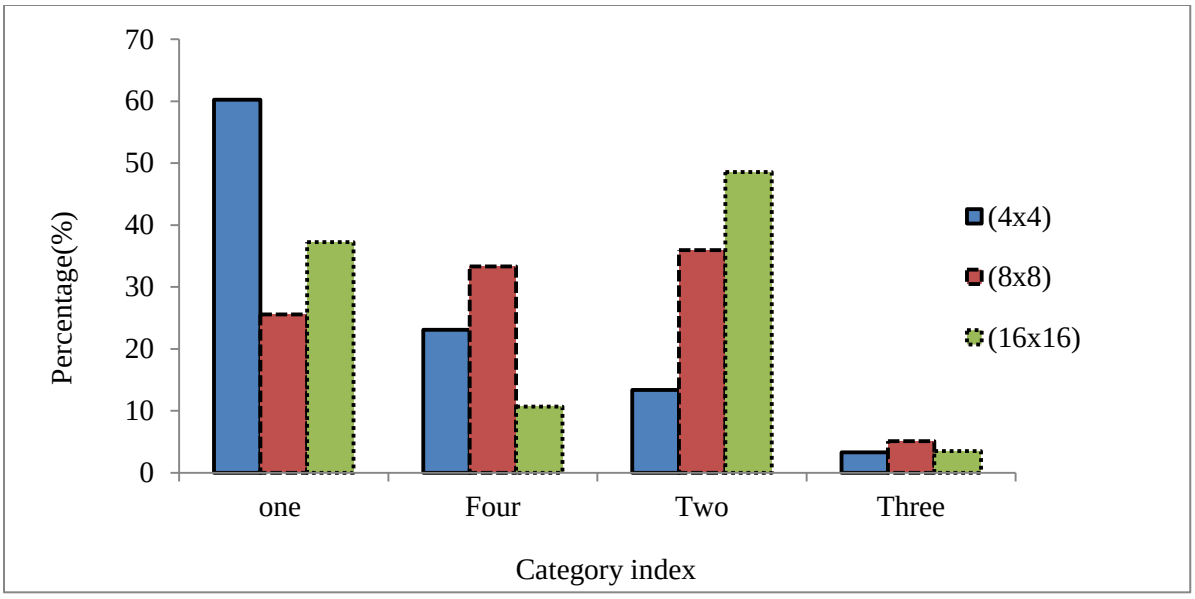


Figure 5.5: Distribution of Motion Vector Categories for Miss America Sequence

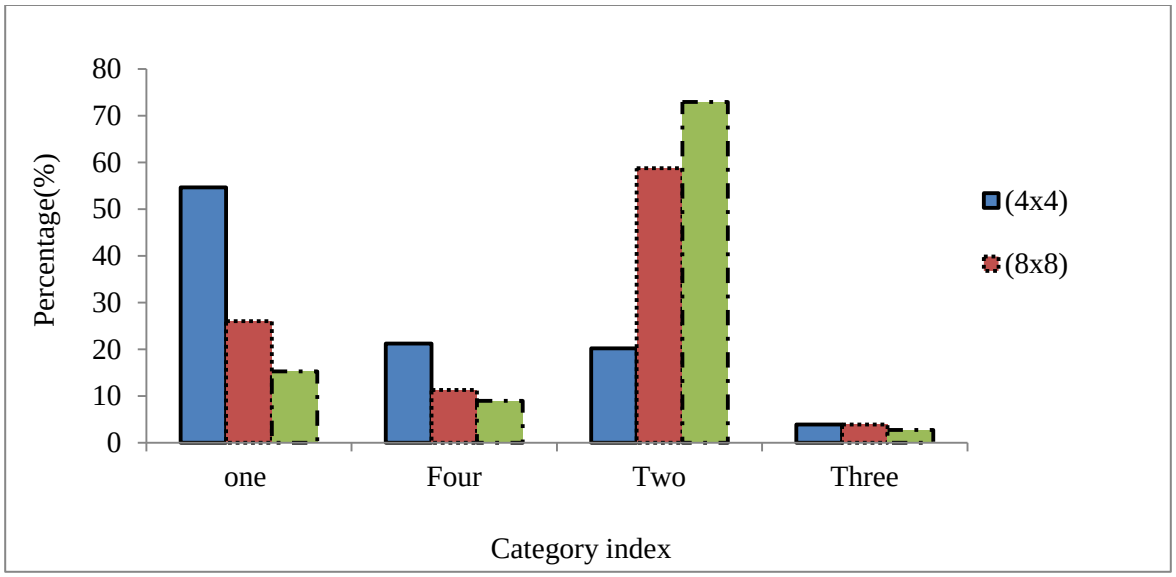


Figure 5.6: Distribution of Motion Vector Categories for Suzie Sequence

Table 5.14: Number of motion vectors versus block size using Miss America test video sequence.

Total number of MVs	First and Forth category MVs	Δ %
647336	528348	81.61882
97748	41932	42.89806
12624	4496	35.6147

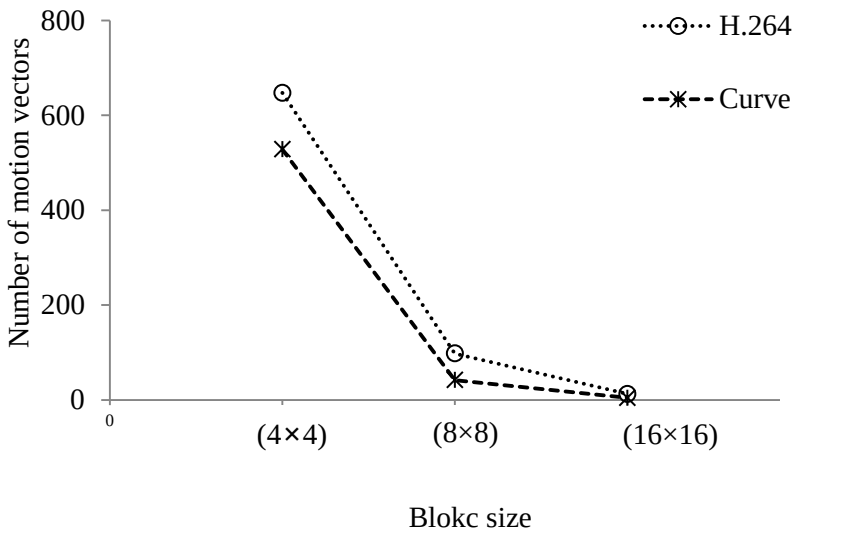


Table 5.14: Number of motion vectors versus block size using Foreman sequence

Total number of MVs	First and Forth category MVs	Δ %
867032	711896	82.10
170600	80808	47.36
30936	10504	33.95

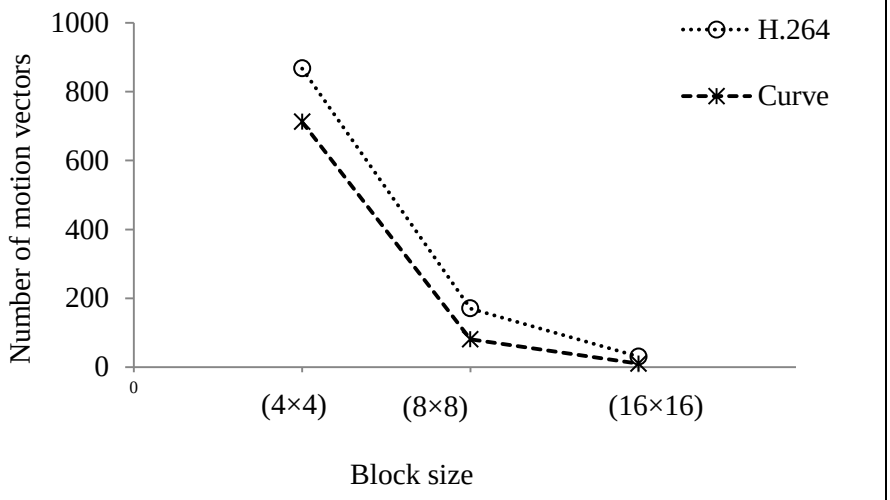


Table 5.15: Number of motion vectors versus block size using Carphone sequence

Total number of MVs	First and Forth category MVs	Δ %
847828	706556	83.33
201336	118592	58.90
47388	22700	47.90

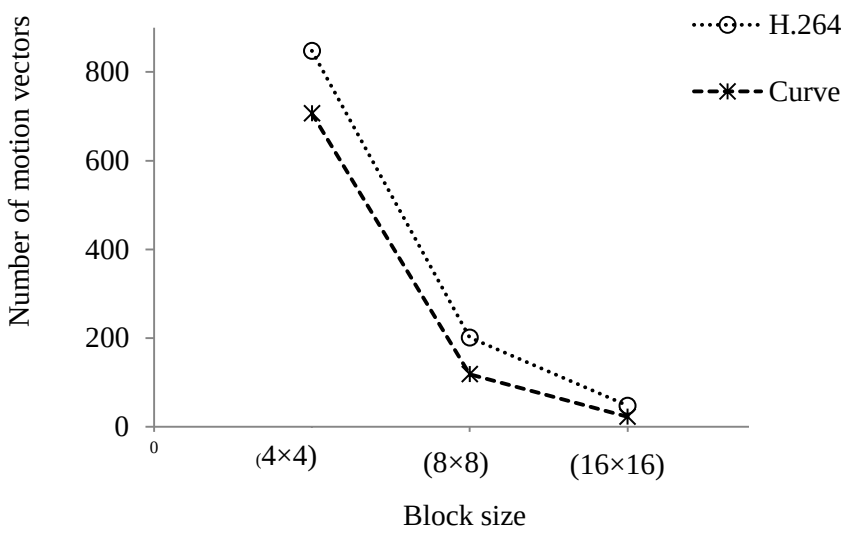
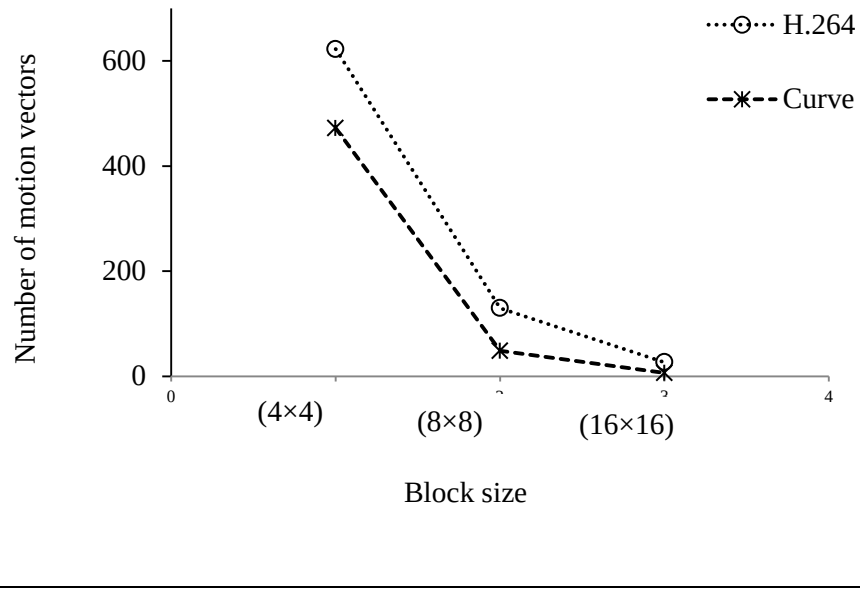


Table 5.16: Number of motion vectors versus block size using Suzie sequence

Total number of MVs	First and Forth category MVs	Δ%
622340	472064	75.85
129568	48388	37.34
26952	6540	24.26



5.3 Bitrate Improvement Evaluation of Proposed Method

The percentage of bitrate saving via encoding motion vectors using our curve fitting algorithm in comparison with H.264 differential motion vector encoding scheme, explained in Chapter 2, has improved by up to 42.7%. The total bitrate differences, Δ(Bitrate), the average PSNR differences, Δ(PSNR), and the coding performances of the proposed scheme in terms of BDBR (Bjontegarred Delsta Bit Rate)[54], are provided in Table 5.18 . The bitrate values comprise the number of bits for encoding residual data and the motion data in both methods. The motion data in the proposed method contains the number of bits required for encoding the curve control points, Category Four motion vectors, and the Category index information while the motion data resulting from the H.264 encoding scheme contains the motion vector differences. The motion vector bitrate is computed as:

$$\Delta(MVBit)\% = \frac{\Delta(MVBit) - \Delta(DCTBit}{MVBit\_H.264}) \times 100 \tag{5.2}$$

where the difference between the number of bits required for encoding motion data and residual data from the proposed method and H.264 is equal to:

$$\Delta MVBit = MVBit\_H.264 - MVBit\_Curve \tag{5.3}$$

$$\Delta DCTBit = DCTBit\_Curve - DCTBit\_H.264 \tag{5.4}$$

The proposed methods provide better results (or a higher percentage of bitrate saving) for larger quantization parameters for all the tested video sequences (Figure 5.10-Figure 5.13). Figure 5.8 shows the average bitrate saving versus the quantization parameter over all the tested video sequences. The bitrate saving follows an increasing trend for the last three larger quantization

parameters while there is a decreasing trend from the smallest quantization parameter to the second smallest quantization parameter. All the result of every single test condition follows the same exact trend as the average bitrate trend shown in Figure 5.8.

Figure 5.9 shows the average bitrate saving versus the block size over all test conditions. The performance of the proposed method increases for smaller block sizes motion estimation methods for two reasons: (1) as it was shown in Section 5.2, the difference between the actual number of motion vectors and the amount of data that should be encoded via our curve fitting algorithm decreases as the block size increases, and (2) larger block size motion estimation generates larger residual data in both systems, AME and H.264 based motion estimation schemes, that itself leads to generating larger differences between residual data bits, Δ(DCTBit). Therefore, this decreases the performance of the proposed motion vector encoding scheme by decreasing Δ(MVBit)% based on Equation (5.2). Therefore, the best performance of the proposed method is achieved for smaller block size motion estimation with higher quantization parameters while for larger block sizes motion estimations and smaller quantization parameters the curve fitting encoding scheme still results in bitrate saving. We should consider that the role of the selected matching blocks is of great importance since the more precise they are chosen the higher bitrate saving is gained from encoding motion vectors via the proposed algorithm.

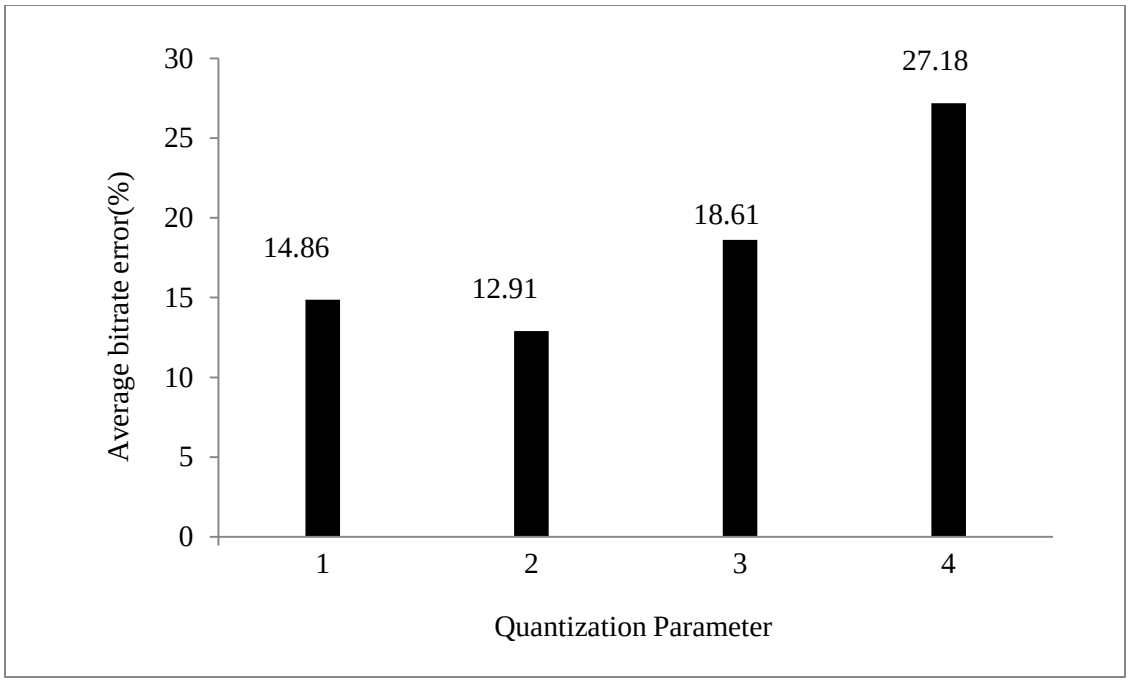


Figure 5.7: Average Motion Vector Bitrate Improvement Versus Quantization parameter

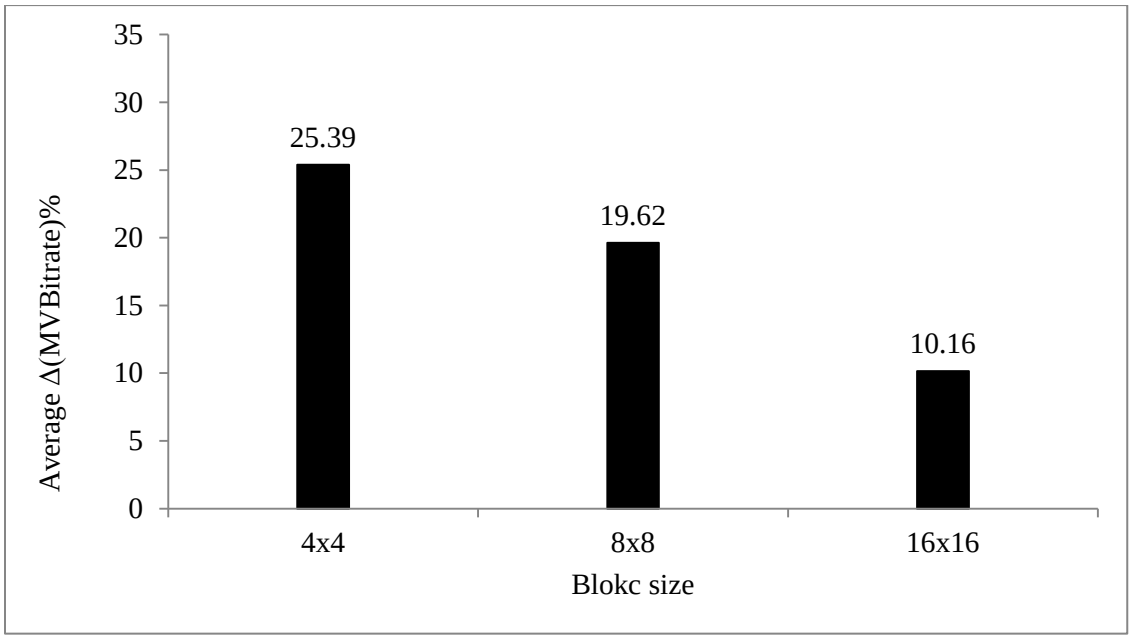


Figure 5.8: Average motion vector bitrate improvement versus Block size

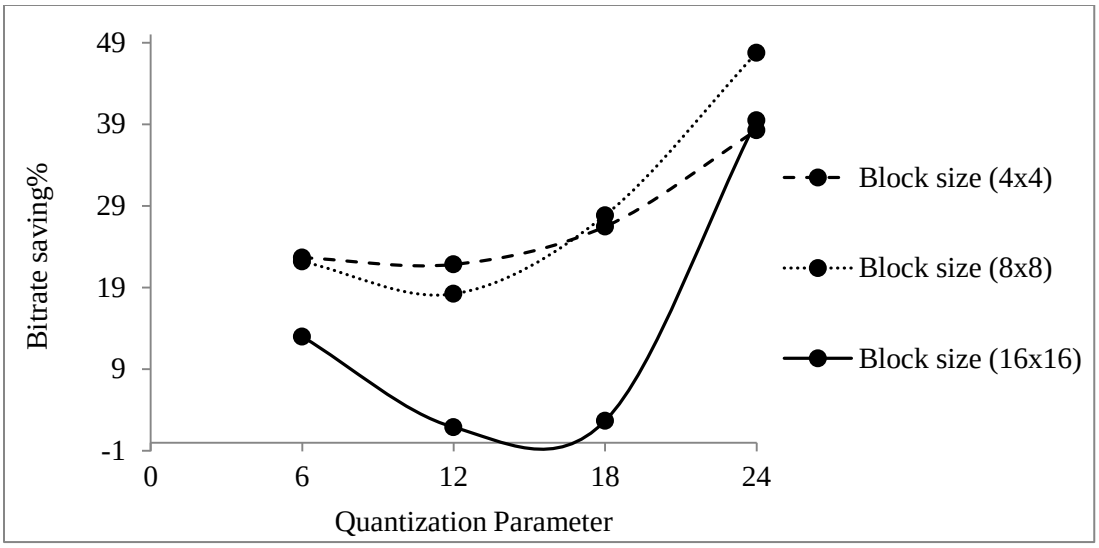


Figure 5.9: Percentage of bitrate saving versus quantization parameter, using Forman sequences

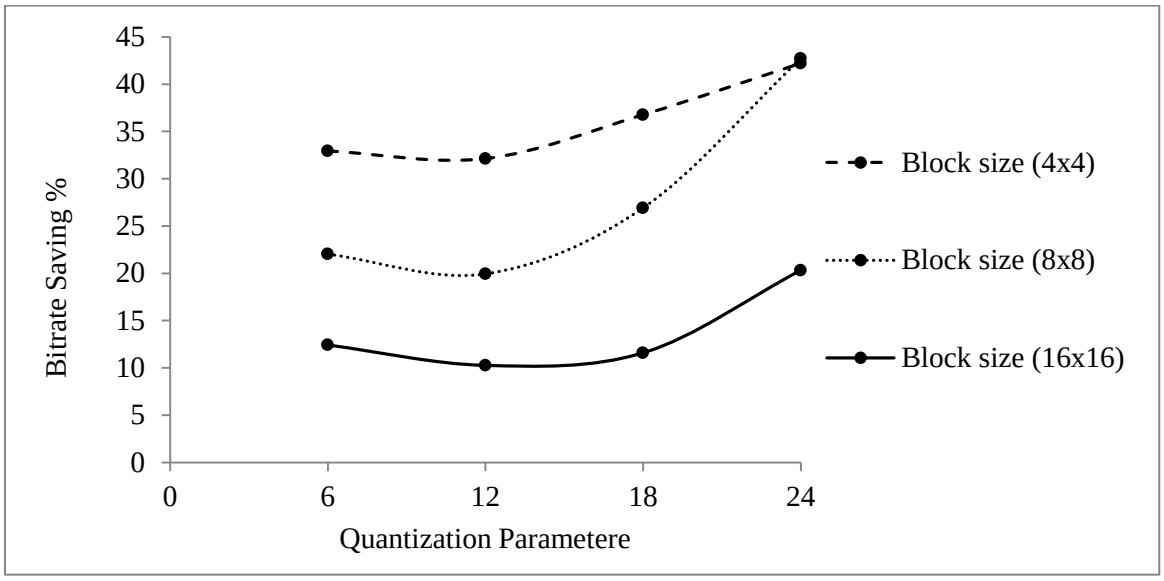


Figure 5.10: Percentage of bitrate saving versus quantization parameter, using Carphone sequences

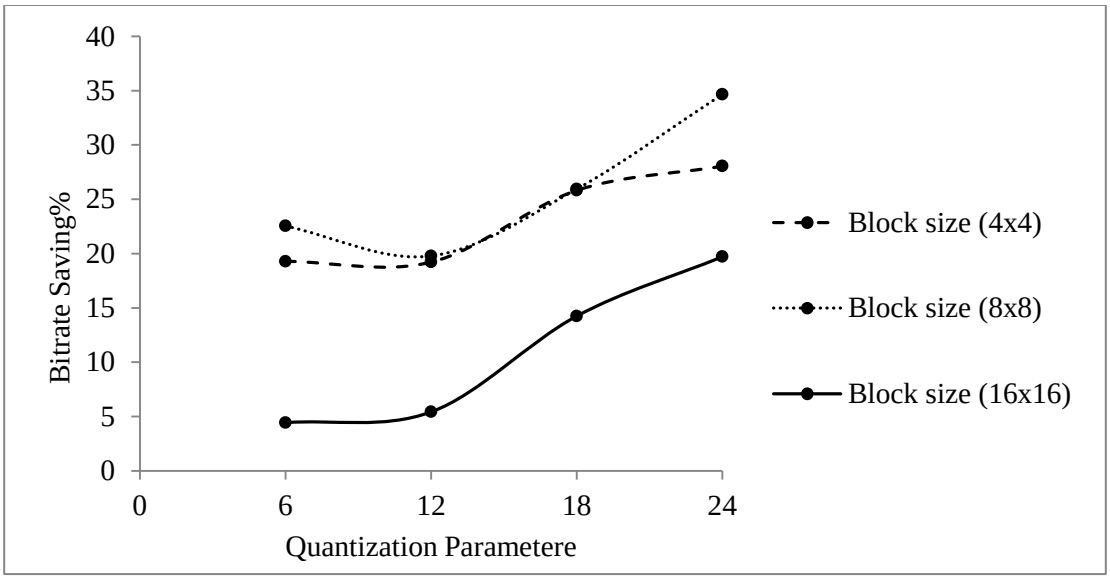


Figure 5.11: Percentage of bitrate saving versus quantization parameter, using Miss America sequences

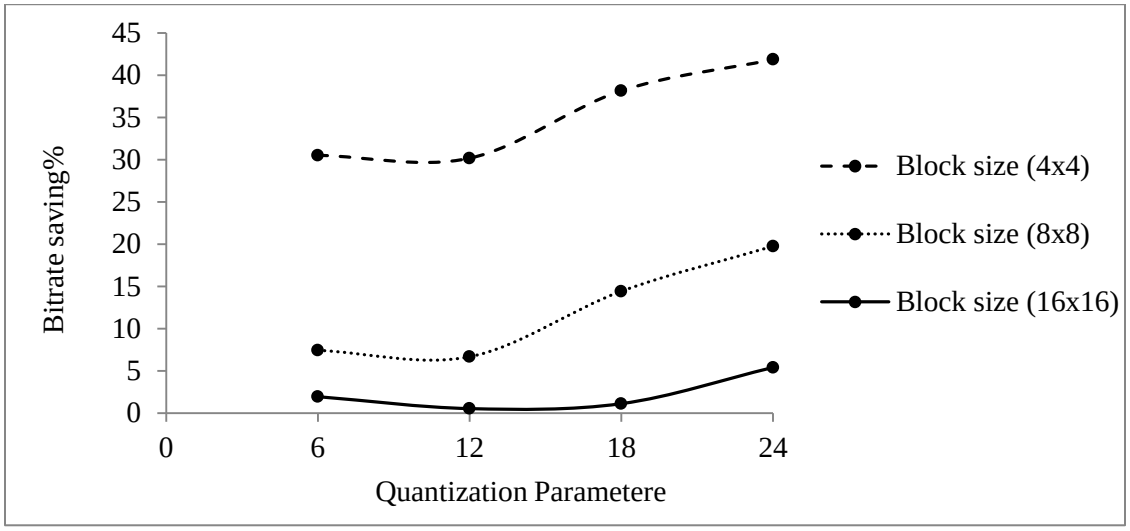


Figure 5.12: Percentage of bitrate saving versus quantization parameter, using Suzie sequences

Table 5.17: Performance of the proposed method in terms of BDBR (Bjontegarred Delta Bit Rate)

Vide Sequence	4x4	8x8	16x16
Foreman	36.3396	33.5782	11.1576
Carphone	48.4641	35.7377	15.4813
Miss America	21.6846	25.9364	28.674
Suzie	41.3587	8.523	3.9947

Chapter 6

Conclusion and Future Directions

6.1 Concluding Remarks

The main objective of the research was to decrease the motion vector bitrate which forms a significant portion of the video bitstream especially in low bitrate communication. Indeed, the previously proposed pre-processing motion vector encoding schemes discussed in the literature are based on generating smaller motion data from motion vectors while for every single motion vector motion data should be computed and encoded. In this research, we have proposed to reduce the amount of motion data by extracting key motion vectors that represent the keypoints of the best fitted curve into the motion vectors. Therefore, the motion vectors are pre-processed in order to reduce the motion data via reducing the number of motion vectors needed to be encoded using a curve fitting algorithm which results in reconstructing the motion vectors conforming to their original values at the decoder side. We showed that selecting the motion vectors of collocated blocks along the video sequence as an independent curve fitting data point sets results in a significant reduction in the number of motion vectors that need to be encoded as the curve keypoints and non-reconstructed motion vectors from the curve especially in smaller

block size motion estimation systems. The subsequent implementation of the motion vector encoding based on the proposed curve fitting algorithm confirmed the bitrate saving for video sequences containing complex and homogenous motion up to 42.7% and 41.84% respectively.

6.2 Future Research Directions

One of the challenges in this work was determining the proper threshold for selecting candidate residual blocks such that encoding motion vectors via curve fitting algorithm reduces the bitrate. Determining the threshold values via the proposed method, discussed in Chapter 4, depends on three factors: block size, quantization parameter, and the type of motion that should be encoded in the video sequence under each test condition. Therefore, a future work on automated threshold determination a strong potential future direction for the current research such that an equivalence between selecting the best motion vector for curve fitting algorithm and residual bitrate increase in comparison with selecting residual block of minimum energy is achieved. Also, as mentioned, the number of motion vectors reduces significantly via mapping them into a smaller set of keypoints and non-reconstructed motion vectors from the curve; however, the increase in the residual data and also header information counteracts the effect of the number of motion vectors that are reduced. Therefore, an optimization on the curve fitting algorithm, in order to reduce the header information as well as an optimum automated threshold determination will result in achieving a higher bitrate saving.

List of References

[1] Golwelkar, A.; Woods, J. W. “Motion-Compensation Temporal Filtering and Motion Vector Coding Using Biorthogonal Filters”. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(4), 417-28, (2007).

[2] Chen, C.-yeh, Chien, S.-yi, Huang, Y.-wen, Chen, T.-chien, Wang, T.-chih, et al. “Analysis and architecture design of variable block-size motion estimation for H.264/AVC”. *IEEE Transactions on Circuits and Systems, 53*(2), 578-593, (2006).

[3] Zhibo C.; Jianfeng X.;Yun H.; Junli Z. “Fast integer-pel and fractional-pel motion estimation for H.264/AVC”. *Journal of Visual Communication and Image Representation*, *17*(2), 264-290, (2002).

[4] Gregory K. Wallace. “The JPEG still picture compression standard”. *Communications of the ACM - Special issue on digital multimedia systems*, *34*(4), 30-44, (1991).

[5] Côté G.; Erol B.; Gallant M.; Kossentini F. H.263 + : Video Coding at Low Bit Rates. *IEEE Transactions on Circuits and Systems*, *8*(7), 849-866, (1998).

[6] *MPEG. "MPEG standards - Full list of standards developed or under development". chiariglione.org.* (2009).

[7] Haskell B.G.; Puri A.; and Netravali A.N. *Digital video: an introduction to MPEG-2*. Chapman & Hall, (1997).

[8] Richardson I.E.G. “H.264 and MPEG-4 Video Compression”, (pp. 45-51). Wiley, (2003).

[9] Malvar H.S.; Staelin D. H., “The LOT: Transform coding without blocking effects,” *IEEE Trans. Accoust., Speech, Signal Processing*, vol.37, 553–559, (1989).

[10] H. S. Malvar, “Biorthogonal and non-uniform lapped transforms for transform coding with reduced blocking and ringing artifacts,” *IEEE Trans. Signal Processing*, vol. 46, pp. 1043–1053, (1998).

[11] Jarske T.; Haavisto P.; Defe’e I. “Post-filtering methods for reducingblocking effects from coded images,” *IEEE Trans. Consumer Electron.*, 521–526, (1994).

[12] H. C. Reeve and J. S Lim, “Reduction of blocking artifacts in image coding,” *Opt. Eng.*, vol. 23, 34–37, (1984).

[13] Meier T.; Ngan K. N.; Crebbin G. “A region-based algorithm for enhancement of images degraded by blocking effects”. *Proc. IEEE Tencon’96*, vol. 1, 405–408, (1996)

[14] Luis A. da Silva Cruz and John W. Woods. “Adaptive motion vector quantization for video coding”. *IEICE of Japan*, *E83-A*(7), 1486-1492, (2000).

[15] Luis A. da Silva Cruz and John W. Woods. “Backward adaptive motion vector VQ for video coding”. *Proceedings of the Picture Coding Symposium* (25-28).

[16] Lee Y.Y. “Motion vector quantization for video coding”. *IEEE Transactions on Image Processing*, 378 – 382, vol.4, (1995).

[17] Joshi, R.L.; Fischer T.R.; Bamberger R.H. “Lossy encoding of motion vectors using entropy-constrained vector quantization”. *IEEE Comput. Soc. Press., Proceedings International Conference on Image Processing*, 109-12, vol.3, (1995).

[18] Yeh J.; Vetterli M.; Khansari M. “Motion compensation of motion vectors”. *IEEE Comput. Soc. Press. Proceedings, International Conference on Image Processing*, 574-7, vol.1, (1995).

[19] ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC Std. (2005).

[20] Heechul Y.; Jungyoup Y.; Kwanghyun W.; Byeungwoo J. “Motion Vector Coding with Decoder Selectable PMV”. *Proceedings of the 2010 2nd International Conference on Signal Processing Systems*, 447-50 vol. 3, (2010).

[21] Kim S.D.; Beom Ra.J. “An efficient motion vector coding scheme based on minimum bit rate prediction”. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society,* 1711-vol. 8, (1999).

[22] Yeh, J. ; Vetterli, M. ; Khansari, M. “Motion compensation of motion vectors”. *Proceedings. International Conference on Image Processing (Cat. No.95CB35819)* (pp. 574-7 vol.1). IEEE Comput. Soc. Press, (1995).

[23] Psannis, K. E. “Motion-based competitive spatio-temporal technique with multi-frames references for efficient H.264/AVC motion information prediction”. *IEEE Int. Symp. Broadband Multimedia Syst. Broadcast., BMSB - Conf. Programme,* (2010).

[24] Tourapis, A.M.; Wo F.; Li S. “Direct mode coding for bipredictive slices in the H.264 standard”. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(1), 119-26, (2005).

[25] Chen, M.C. ; Willson, A.N. Jr. “A spatial and temporal motion vector coding algorithm for low-bit-rate video coding”. *IEEE Comput. Soc. Proceedings. International Conference on Image Processing*, 791-4 vol.2, (1997).

[26] Laroche, G.; Jung j.; Pesquet-Popescu B. “A spatio-temporal competing scheme for the rate-dstortion optimized selection and coding of motion vectors”. *14th European Signal Processing Conference*. (2006).

[27] Yang J.; Won K.; Yung-Lyul Lee Y.-L. ; Jeon B. “Motion vector coding using optmal prediction”. *Proceedings of the 2009 16th IEEE International Conference on Image Processing*, 1033-6, (2009).

[28] Golomb S.W. “Run-length encoding”. *IEEE Trans. on Inf. Theory*, 399-401, (1966).

[29] Marpe D.; Schwarz H.; Wiegand T. “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard”. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(13), 620-7, (2003).

[30] Farm G. “Curves and Surfaces for Computer Aided Geometric Design*”: A practical Guide*. Academic Press,(1996).

[31] Mettke, H. “Convex cubic Hermite-spline interpolation”. *Journal of Computational and Applied Mathematics*, *11*(3), 377-8, (1994).

[32] Iyengar, S.R.K.; Jain, P. “Spline finite difference methods for singular two point boundary value problems”. *Numerische Mathematik*, *50*(3), 363-76, (1987).

[33] Boor da B.; Schoenberg I. J. “Cardinal interpolation and spline functions VIII. The Budan-Fourier Theorem for splines and applications”. *Applied and Computational Harmonic Analysis*, 1-79, vol. 501(1976).

[34] Yuksel C.; Schaefer S. ; Keyser J. “Parameterization and applications of Catmull-Rom curves”. *Computer Aided Design*, *43*(7), 747-55, (2011).

[35] Catmull E.; Rom R. “A class of local interpolating splines. In Computer Aided Geometric. Design, R. E. Barnhill and R. F. Reisenfeld, Eds. Academic Press, New York, 317–326, (1976).

[36] Karczewicz M.; Kurceren R.; “A proposal for SP-frames, ITU-T SG16/6 document VCEG-L27”, Eibsee, Germany, (2001).

[37] Cover T.M.; Thomas J.A. "*ELEMENTS OF INFORMATION THEORY"*. Wiley (2006).

[38] Hang H.-ming; Chen, J.-jone. "Source Model for Transform Video Coder and Its Application". *IEEE Transactions on Circuits and Systems*, *7*(2), 287-298, (1997).

[39] Ma S.; Gao W.; Lu Y. "Rate-Distortion Analysis for H.264/AVC Video Coding and its Application to Rate Control". *IEEE Transactions on Circuits and Systems*, *15*(12), 1533-1544, (2005).

[40] Ribas-corbera J.; Lei S. "Rate Control in DCT Video Coding for Low-Delay Communications". *IEEE Transactions on Circuits and Systems*, *9*(1), 172-185, (1999).

[41] Lee H.-ju; Chiang T.; Zhang Y.-qin. "Scalable Rate Control for MPEG-4 Video". *IEEE Transactions on Circuits and Systems*, *10*(6), 878-894, (2000).

[42] Vetro A.; Sun H.; Wang Y. "MPEG-4 Rate Control for Multiple Video Objects". *IEEE Transactions on Circuits and Systems*, *9*(1), 186-199, (1999).

[43] Makai B.; Engelhardt T.; Mehlan R. "A New Rate Control Scheme Using Quadretic Rate Distortion Model". *IEEE Transactions on Circuits and Systems*, *7*(1), 246-250, (1997).

[44] Tian L.; Sun Y.; Zhou Y.; Xu X. "ANALYSIS OF QUADRATIC R-D MODEL IN H.264/AVC VIDEO CODING". College of Computer Science & Engineering, University of Electronic Sci. & Tech. of China Department of Computer Science , University of Central Arkansas , USA. *Image Processing*, 2853-2856, (2010).

[45] Lin L.-jin; Ortega A.; Kuo C.J. "Rate Control Using Spline-Interpolated R-D Characteristics". *SPIE 2727, 111*, Vol. 2727, pp. 111-122, (1996).

[46] Webb J.L.H.; Oehler K. “A simple rate-distortion model, parameter estimation, and application to real-time rate control for DCT-based coders”. *Proceedings of International Conference on Image Processing*, Vol. 0, pp. 13-16, (1997).

[47] He Z.; Mitra S.K.; Fellow L. “Optimum Bit Allocation and Accurate Rate Control for Video Coding via -Domain Source Modeling”. *IEEE Transactions on Circuits and Systems*, *12*(10), 840-849, (2002).

[48] He Z.; Mitra S. K. “A Unified Rate-Distortion Analysis Framework for Transform Coding”. *IEEE Transactions on Circuits and Systems*, 11(12), pp. 46-49, (2001).

[49] Kim Y. K.; He Z.; Matra S. K. “A novel linear source model and a unified rate control algorithm for h.263 / mpeg-2 / mpeg-4”. *IEEE International Conference on Acoustics, Speech, and Signal Processing,. Proceedings*, pp. 1777-1780, (*2001*).

[50] He Z.; Kim Y. K.; Mitra S. K.; “Low-Delay Rate Control for DCT Video Coding via ρ-domain source modeling”. *IEEE Transactions on Circuits and Systems*, *11*(8), 928-940, (2001).

[51] Shen M.; Kuo C.J. “Rate Control for H.264 Video with Enhanced Rate and Distortion Models”. *IEEE Transactions on Circuits and Systems*, *17*(5), 517-529, (2007).

[52] Liao K.-ying; Yang J.; Sun M. “Rate-Distortion Cost Estimation for H.264/AVC”. *IEEE Transactions on Circuits and Systems for Video Technology*, *20*(1), 38-49, (2010).

[53] Z. G. LI, F. Pan, K. P. Lim and G. N. Feng, “Adaptive Basic Unit Layer Rate Control for JVT,” Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, JVT-G012, Pattaya, Thailand, pp. 7-14, Mar. 2003.

[54] G. Bjøntegaard, “Calculation of average PSNR differences between RD-curves,” ITU-T SG16/Q6 Doc. VCEG-M13, April 2001.